IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 January 2023, accepted 29 January 2023, date of publication 1 February 2023, date of current version 8 February 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3241753

RESEARCH ARTICLE

Efficient Adaptive Monte Carlo Uncertainty Forecasting for High Dimensional Nonlinear Dynamic Systems

ANDREW W. VANFOSSEN AND MRINAL KUMAR, (Member, IEEE)
Department of Mechanical and Aerospace Engineering, The Ohio State University, Columbus, OH 43210, USA

Corresponding author: Andrew W. VanFossen (vanfossen.24@osu.edu)

This work was supported by the Air Force Office of Scientific Research under Grant FA9550-20-1-0083. The doctoral work of Andrew W. VanFossen was also supported by The Ohio State University.

ABSTRACT The problem of uncertainty forecasting for complex dynamical systems in the framework of particle methods can be effectively addressed through the solution methodology known as adaptive Monte Carlo (AMC). Monte Carlo (MC) methods involve discretizing the initial probability density function (pdf) followed by forward propagation of particles through system dynamics to obtain an approximate particle representation of the evolved state uncertainty. While simple to implement, MC faces questions surrounding transient statistical consistency and rate of convergence. AMC (adopted here) addresses these issues on-the-fly using defined bounds on estimation accuracy alongside ensemble enrichment routines. This paper presents several improvements in the ensemble enrichment module of AMC. The AMC platform is re-engineered to include the novel implementation of a parallel global stochastic optimization routine in conjunction with additional module enhancements that work together towards the goal of efficient forecasting. Moreover, the efficacy of algorithms utilized within every submodule of AMC are detailed and improved upon under the framework of arithmetic minimization and parallelization. Each submodule is profiled to determine computational bottlenecks, optimized or replaced with more efficient methods such as simulated annealing (SA), and parallelized ultimately leading to a clock time reduction of $200 - 400\%$ for benchmark entry descent and landing, Lorenz-96, and Lotka-Volterra models.

INDEX TERMS Adaptive Monte Carlo, arithmetic minimization, parallel algorithms, nonconvex optimization, simulated annealing, uncertainty quantification.

I. INTRODUCTION

Numerical simulations have served as an important tool for a wide variety of disciplines, including engineering, mathematics, physics, biology, and economics, among others. Simulations have become the preferred alternative to expensive experimentation as well as when real-life systems are too difficult to study analytically. Advances in simulation methodologies, software availability, sensitivity analysis, and stochastic optimization have combined to make numerical simulations one of the most widely accepted and used tools in system analysis and operations research [1]. Simulations are an appropriate tool when simple mathematical formulations are untenable due to complex system dynamics, analytical techniques do not apply or are too cumbersome, the interaction among system variables play an important role, one wishes to gain insight into system behavior under novel circumstances, and/or expanded or compressed system timelines are of interest. However, without proper care, simulations can be inexact with implementation variability pertaining to the intrinsic randomness of nature, data and parameter uncertainty, model uncertainty: including necessary simplifications, numerical inaccuracies, and others [2]. Consequently, the reliability of model outcomes and predictive analysis of uncertainties propagated throughout numerical simulations is highly important. There are several

The associate editor coordinating the review of this manuscript and approving it for publication was Thomas Canhao Xu.

ways to ascertain the validity of a model, e.g., reexamining the problem formulation, checking the mathematical consistency of expressions, varying input parameters to study model behavior, and comparing simulation outputs to historical data. Once a model has been constructed, analytical or numerical solution methods can be implemented to obtain exact or approximate solutions to the system at hand [1]. This paper focuses on a numerical solution method called Monte Carlo simulations (MCS) or stochastic computer simulations towards the goal of predictive uncertainty quantification (UQ). UQ can be defined as the process of characterizing the uncertainties of one's lack of knowledge pertinent to physical reality. More specifically, the forward propagation of uncertainty that forecasts model output by propagating random initial conditions through a stochastic dynamic system is the central problem studied in this paper.

The Fokker-Planck equation (FPE) (see Sec.(II)), captures the evolution of the state-pdf (uncertainty) of nonlinear dynamical systems perturbed by white noise. Its solution is often sought in the fields of structural mechanics, fluids, chemical processes, control theory, and prognostics, among others. Closed-form analytical solutions to the FPE only exist for a handful of systems. For the case involving a linear system with Gaussian initial uncertainty, the solution to the FPE can be found by solving the corresponding matrix Riccati differential equation. Traditional discretization techniques such as the finite difference method or finite element method exist with more general applicability. Unfortunately, the *curse of dimensionality* limits the use of such methods in providing accurate solutions of the FPE for relatively high dimensional systems [3]. For reference, a nonlinear dynamic system with six states can be thought of as ''high dimensional'' for the FPE [4]. Approaches such as the partition of unity finite element method (PUFEM), particle-PUFEM and tensor decomposition have also been implemented in an effort to curtail the exponential growth in the number of unknowns associated with the curse of dimensionality. For specifics on these algorithms and their implementations, please see [5], [6], [7], [8], [9], [10], [11], [12], and [13].

An alternative numerical approximation to solving the FPE is offered by the MC suite of algorithms. MC algorithms have been implemented for a wide class of dynamic systems due to their simplicity, flexibility, parallelizability, and provable asympotic convergence. MC creates a particle representation of the evolved state-uncertainty, which can in turn be utilized in computing desired statistics or approximating the state-pdf. Discretization in MC is achieved by randomly drawing a ''sufficiently large'' number of samples from the underlying probability space. Each particle is forward propagated through the system dynamics, which results in a new ensemble at future time t. Similar to initial time t_0, desired statistics or the state-pdf can be computed utilizing the propagated ensemble. It is known that MC estimation converges at a rate $O(n^{-1/2})$ where n is the ensemble size and is notably independent of the dimension of the problem. Because of this, MC is capable of circumventing the curse of dimensionality that plagues common numerical solutions of the FPE mentioned previously. It should be noted improving the computational complexity of MC is an active area of research and significant savings have been achieved using multilevel (ML) methods, multi-fidelity (MF) models, or polynomial maps [14], [15], [16], [17], [18], [19], [20], [21]. That is, the computational benefits associated with substituting sample propagation with polynomial evaluations allows one to rapidly propagate an ensemble of particles [15]. Multilevel methods utilize coarsened discretizations of the governing equations to restrict low-fidelity models whereas multi-fidelity methods generalizes the types of permitted models [14], [16]. In essence, MLMC and MFMC determine closed form expressions for resource allocation that significantly improve the performance of traditional MC. Another generalized approach known as approximate control variates further unifies the MLMC and MFMC based methods that also shows significant improvements through the optimization of resource allocations [20].

In traditional MC, the size of the ensemble (n) remains fixed as it propagates through the state-space. It has been shown in [22] that a fixed sized ensemble does not represent the underlying state-pdf with equal accuracy at all times. In other words, while the initial ensemble may represent the initial state-pdf with desired accuracy, the propagated ensemble may be an under- or, even over-sampling of the true instantaneous state-pdf at different times. This translates to the difficult and yet unsolved problem of determining a-priori how large a ''sufficiently large'' initial ensemble must be to represent the state-pdf with desired accuracy at all times. To address the aforementioned, this paper utilizes an adaptive MC platform developed in [23]. The AMC platform is built on a closed-loop architecture towards the goal of controlling the transient forecasting performance and associated computational cost. AMC quantifies its forecasting performance in terms of the estimation of application specific quantities of interest (QoIs) as defined in Sec.(III-A). The platform allows the user to prescribe upper and lower bounds on QoI forecasting error at the front end. When the QoI forecasting error exceeds the user-prescribed upper bound, an *ensemble enhancer* scheme is activated that adds appropriately chosen new particles to the ensemble [23]. When the QoI forecasting error is measured to be less than the user-prescribed lower bound, AMC executes an *ensemble thinner* scheme that identifies particles in the current ensemble to be halted in the interest of reducing computational load. Selection of particles for ensemble thinning is made on the basis of particles' relative current state-pdf value. On the other hand, the theoretical basis for ensemble enhancement lies in the Koksma-Hlawka inequality (Sec.(III-A)), which allows QoI error to be held within bounds by identifying new particles that minimize the discrepancy of the initial ensemble with respect to the initial state-pdf. While it lies at the heart of the enhancement routine, the optimization of discrepancy is difficult due to challenges linked to high-dimensionality and non-convexity. As a result, [23] did not directly perform the optimization of discrepancy to identify new particles and

instead employed indirect means to reduce initial ensemble discrepancy. This was done through a routine known as *admissible intervals* by sampling subsections of the domain that enforce space-filling and non-collapsing criteria [24]. The current paper restructures the AMC ensemble enhancer by devising a novel parallelized stochastic global optimization method for the discrepancy function, based on SA. This allows ensemble enhancement to be better adapted to the Koksma-Hlawka inequality, while also achieving notable improvements to computational efficiency as a result of the parallelized implementation.

A. CONTRIBUTIONS/SIGNIFICANCE OF WORK

This paper makes two main contributions to the AMC platform for uncertainty forecasting. First, it addresses the ''indirect discrepancy minimization'' issue of the existing AMC platform [23]. As mentioned above, the existing AMC platform merely employs efficient sampling techniques based on space-filling and non-collapsing criteria to indirectly achieve a reduction in discrepancy. In this paper, a novel parallelized global stochastic optimization approach is devised that tackles the problem of high-dimensionality and non-convexity in the minimization of discrepancy. The global optimization scheme is based on SA, which in its native form, is a sequential algorithm. This paper develops a parallelized version of SA that leverages the exploration history of parent CPUs to determine optimal candidates of the $U_0 + i^{\text{th}}$ ensemble as defined by successive *new* discrepancy cost functions. That is, for m available computational processors parallel SA determines m additional particles to join U_0 through the direct optimization of distinct discrepancy cost functions on each parallel CPU. This is accomplished by children CPUs utilizing *near optimal* estimates of candidate particles up to the $U_0 + (i-1)^{\text{th}}$ ensemble, which are continually updated throughout SA temperature cycles. The result is the addition of m particles under the same clock time as sequential SA (substantially quicker than *admissible intervals*) with no notable degradation in the quality of the overall ensemble.

Second, this paper investigates computational bottlenecks within the existing AMC platform, examines the architecture of each algorithm, implements arithmetic minimization and parallelization where possible, and benchmarks the improvements against complex high dimensional dynamic systems. It is shown through numerical studies that these architectural changes lead to a $200-400\%$ improvement in computational efficiency in non-trivial uncertainty forecasting problems up to $N = 30$ dimensional space. This is accomplished through minimizing the number of arithmetic operations required in sub-modules, restructuring AMC to reduce memory overhead, and exploiting efficient mathematical libraries. That is, while modules such as the ensemble enhancer are restructured via parallel SA, the memory overhead and computational burden of underlying operations in each module of AMC are also profiled for further refinement and improvement.

Ultimately, this manuscript highlights deficiencies in the current state-of-the-art AMC platform and restructures modules to directly address the optimization of discrepancy, which in conjunction with arithmetic minimization and parallelization substantially reduces the computational burden of AMC. The significance of implementing an optimized, parallel architecture for the AMC platform is in providing trustworthy, actionable, and timely intelligence to decision making entities. The ability to achieve timely forecasts with user-prescribed accuracy is crucial for applications with very low margin for error, e.g., prediction of probability of collision in space, prescriptive maintenance of high value assets, etc. The rest of the paper is organized as follows: in Sec.(II) the general uncertainty forecasting problem as well as pertinent dynamic systems are introduced. The AMC platform is discussed in Sec.(III). Specific modules and components are also analyzed and re-engineered in Sec.(IV). Discussions connected to parallelization are detailed in Sec.(V). Numerical results for the entry, descent, and landing, Lorenz-96, and Lotka-Volterra systems are shown in Sec.(VI). Finally, conclusions are drawn in Sec.(VII).

II. PROBLEM STATEMENT

This paper concerns the application of the AMC platform to the nonlinear dynamic system with initial condition uncertainty and random excitation given by the following stochastic differential equation (SDE) [25]:

$$d\mathbf{x} = \mathbf{f}(t,\mathbf{x})dt + \mathbf{g}(t,\mathbf{x})d\mathbf{B}(t), \qquad \mathbf{x}_0 \sim \mathcal{W}_0(\mathbf{x}) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$ denotes the system state and $\mathbf{x}_0$ the initial condition with associated pdf $\mathcal{W}_0$. The ''process noise'' term, $d\mathbf{B}(t)$, is an M-dimensional Brownian motion term with zero mean and correlation function $Q\delta(t_1 - t_2)$. The nonlinear vector function $\mathbf{f}(t,\mathbf{x}) : [0,\infty) \times \mathbb{R}^N \to \mathbb{R}^N$ corresponds to the deterministic part of the system and $\mathbf{g}(t,\mathbf{x}) : [0,\infty) \times \mathbb{R}^N \to \mathbb{R}^{N\times M}$ is a nonlinear matrix noise-influence function. For stochastic systems given in (1), the corresponding FPE governing the time propagation of the state-pdf $\mathcal{W}(t)$ is given by:

$$\begin{aligned}\frac{\partial}{\partial t}\mathcal{W}(t,\mathbf{x}) &= \mathcal{L}_{\mathcal{FP}}[\mathcal{W}(t,\mathbf{x})], \quad \mathbf{x}_0 \sim \mathcal{W}(t_0,\mathbf{x}) \\ &= \left[-\sum_{i=1}^{N}\frac{\partial}{\partial x_i}D_i^{(1)}(\cdot) + \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial^2}{\partial x_i \partial x_j}D_{ij}^{2}(\cdot)\right] \\ D^{(1)}(t,\mathbf{x}) &= \mathbf{f}(t,\mathbf{x}), \quad D^{(2)}(t,\mathbf{x}) = \frac{1}{2}\mathbf{g}(t,\mathbf{x})Q\mathbf{g}^{\top}(t,\mathbf{x})\end{aligned} \tag{2}$$

where $D^{(1)}$ and $D^{(2)}$ are known as the drift coefficient vector and diffusion coefficient matrix, respectively. For the current article, the focus is on the case where process noise is absent, thereby reducing (1) to:

$$d\mathbf{x} = \mathbf{f}(t,\mathbf{x})dt, \qquad \mathbf{x}_0 \sim \mathcal{W}_0(t_0,\mathbf{x}) \tag{3}$$

and the time evolution of the state-pdf $\mathcal{W}_0(t_0,\mathbf{x})$ shown in (3) is given by the stochastic Liouville equation:

$$\frac{\partial}{\partial t}\mathcal{W}(t,\mathbf{x}) = \mathcal{L}[\mathcal{W}(t,\mathbf{x})] = -\sum_{i=1}^{N}\left(f_i\frac{\partial \mathcal{W}}{\partial x_i} + \mathcal{W}\frac{\partial f_i}{\partial x_i}\right) \tag{4}$$

where $\mathcal{L}(\cdot)$ is the stochastic Liouville operator. Since the solution of (4) must be a valid pdf the following conditions are also needed: $\lim_{x\to\infty} \mathcal{W}(t, \mathbf{x}) = 0$ and $\int_{\mathbb{R}^N} \mathcal{W}(t, \mathbf{x})d\mathbf{x} = 1, \forall\, t \in [0, \infty)$. In MCS, realizations of initial uncertainty are generated via random sampling $\{\mathbf{x}_0^i\}_{i=1}^n \sim \mathcal{W}_0$, where n denotes the total number of particles in the ensemble. Each particle is forward propagated in time through system dynamics to obtain an approximate representation of the evolved state-pdf. That is, $\{\Phi_t(\mathbf{x}_0^i)\}_{i=1}^n \propto \mathcal{W}_t(\mathbf{x})$ where $\Phi_t(\cdot)$ is the system dynamics map that maps initial conditions to the current state. For noise driven dynamic systems, (1), the map Φ_t is related to the strong form solution of the SDE, given in the form of an Itô integral [25]. For dynamics systems with no process noise, (3) Φ_t is the state-transition function $\mathbf{x}(t) = \Phi_t(\mathbf{x}_0)$ [26]. While MCS holds for general nonlinear systems ($\mathbf{f}$), this paper will consider three representative system models that capture nonlinearity and dimensionality related complexities, namely, entry, descent, and landing (5), the Lorenz-96 system (6), and the Lotka-Volterra (7) system.

A. ENTRY, DESCENT, AND LANDING

The Vihn's equations are simplified hypersonic entry, descent, and landing (EDL) dynamics under the assumptions of purely longitudinal dynamics and a nonrotating spherical planet with zero-bank angle flight [27]. Given in nondimensional form, the governing equations are as follows:

$$\dot{h} = V \sin\gamma \tag{5a}$$

$$\dot{V} = -\frac{\rho R_0}{2B_c}V^2 - \frac{gR_0}{v_c^2}\sin\gamma \tag{5b}$$

$$\dot{\gamma} = \frac{\rho R_0}{2B_c}\frac{C_L}{C_D}V + \frac{gR_0}{v_c^2}\cos\gamma\left(\frac{V}{1+h} - \frac{1}{V}\right) \tag{5c}$$

where nondimensionalized altitude, nondimensionalized planet-relative speed, and flight-path angle correspond to the three states (h, V, γ). In this paper, the central object will be assumed to be Mars, whereby $R_0 = 3.397 \times 10^6$ m. The ballistic coefficient (B_c) in (5) is set at $B_c = 72.8$ kg/m^2, the lift-to-drag ratio $(C_L/C_D) = 0.3$, $g = 3.721$ m/s^2, and $v_c = \sqrt{\mu/R_o}$ where $\mu = 4.282837 \times 10^{13}$ m^3/s^2. The dynamics defined in (5) provide AMC with a relatively complex low-dimensional use case for parallelization within AMC. That is, for strict error control scenarios, (5) is expected to trigger AMC ensemble adaptations throughout the simulation's duration. Utilizing (5) here also provides the parallel AMC platform with comparison points found in [28] and [23] for baselines before moving to higher dimensional systems defined by the Lorenz-96 and Lotka-Volterra models in the succeeding sections.

B. LORENZ-96 MODEL

The Lorenz-96 model is a high dimensional forced dissipative system with quadratic nonlinear terms and expands upon Lorenz's simpler 3D models, such as Lorenz-63 and Lorenz-84. The state variables (x_k in (6)) can be interpreted as an atmospheric quantity such as temperature, pressure or vorticity, measured along a circle of constant latitude [29]. That is, the latitude circle is divided into N equal sectors, with a distinct x_k variable for each sector such that the index $k = 1, \ldots, N$ indicates the longitude and thereby describes waves in the atmosphere [30].

$$\dot{x}_k(t) = -x_{k-2}(t)x_{k-1}(t) + x_{k-1}(t)x_{k+1}(t) - x_k(t) + F \tag{6}$$

The value of (6) lies in the ease of implementation for simulation purposes while at the same time potentially exhibiting complex dynamics for suitable choices of N and F. Typical state-space sizes (N) found throughout literature are: 4, 8, 36, and 40, with applications in chaotic attractors, model error, predictability, and data assimilation [29], [31], [32], [33]. For $F > 0$ the equilibrium $x_F = (F, \ldots, F)$ exhibits several Hopf or Hopf-Hopf bifurcations for $N \geq 4$. More so, the first Hopf bifurcation is always super critical, which implies the birth of a stable periodic attractor [30]. For $F < 0$, the dynamics of (6) are influenced by symmetry, meaning bifurcation points depend on the state-space dimension N. For odd dimensions, the first bifurcation of the equilibrium x_F is a supercritical Hopf bifurcation. In even dimensions, symmetry causes the occurrence of a pitchfork bifurcation for the equilibrium x_F and the resulting stable equilibria can exhibit a pitchfork bifurcation again [30]. For $F = 0$, the equilibrium x_F is stable in any dimension. It should also be noted that stable equilibria in (6) eventually lose stability through a supercritical Hopf bifurcation for both $F > 0$ and $F < 0$ giving birth to a periodic orbit [30]. Further bifurcations of the stable periodic orbit can also be observed, which may result in the birth of a chaotic attractor. Scenarios visualizing 2 of the 4 dimensions for a 4D Lorenz-96 system are shown in Fig.(1). As the forcing parameter (F) increases from 0.5, the Lorenz-96 system bifurcates, branches, and eventually evolves into chaos when $F = 12.5$. As such, identical QoIs related to the chaotic Lorenz-96 attractor are inherently more difficult to estimate than those exhibiting more stable behavior. Restated, a chaotic Lorenz-96 system with identical QoIs and accuracy thresholds to that of a more stable attractor will require a larger particle ensemble within AMC. This phenomenon is reinforced in Sec.(VI). Additional scenarios that explore the state-space dimension N, the forcing parameter F of (6), performance improvements gained through parallelization, as well as complexities associated with the system dynamics and QoI are analyzed in Sec.(VI).

C. COMPETITIVE LOTKA-VOLTERRA MODEL

Equation(7) below models the interaction of species and was introduced independently by Volterra in 1931 and by Lotka in 1925:

$$\frac{dx_i}{dt} = r_i x_i\left(1 - \sum_{j=1}^{N}\alpha_{ij}x_j\right) \tag{7}$$

where x_i denotes the population size of the i^{th} species relative to its carrying capacity K_i (uniform), r_i the inherent per-capita growth rate, and α_{ij} are intraspecific (if $i = j$) or interspecific (if $i \neq j$) interaction coefficients. In this paper, we set $\alpha_{ij} \geq$

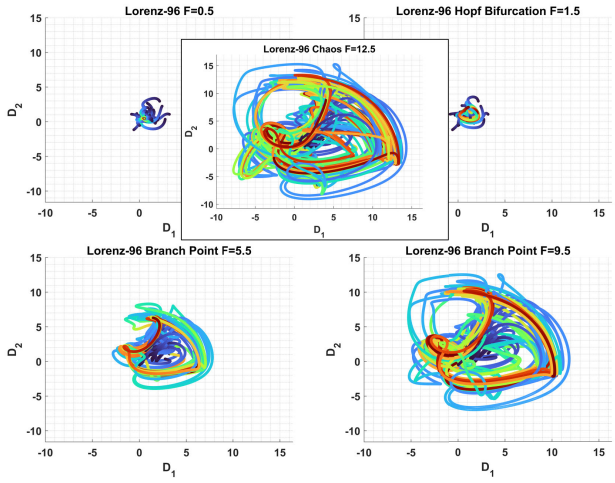

FIGURE 1. Lorenz-96 Route to Chaos.

0, which corresponds to each species competing with one another. More so, the intraspecific competition terms are set to unity ($\alpha_{ii} = 1$ for all $i = 1, \ldots, N$) due to x_i being expressed in terms of the carrying capacity. The condition $r_i > 0$ will also be invoked to ensure that all solutions with non-negative initial conditions remain bounded and asymptotically approach the region $0 \leq x_i \leq 1$ [34]. The Lotka-Volterra model is usually a starting point for natural processes that exhibit aperiodic fluctuations and is of interest here because (7) can easily be extended to a high-dimensional system by increasing N. Equation(7) also offers a juxtaposition to Lorenz-96 in which high dimensional systems can be explored along with their behavior as strange attractors or as they evolve into chaos. Specifically, for three or fewer interacting species (7) cannot have chaotic solutions [35], [36]. However, for five or more species any of the above described types of dynamic behavior can occur, which will be the focus of this paper [37]. As the computational burden of the AMC platform depends on the complexities associated with the system dynamics as well the QoI, it is important to study both (7) and the Lorenz-96 system (6) when highlighting potential performance gains.

III. ADAPTIVE MONTE CARLO

This section describes the existing form of the AMC uncertainty forecasting platform as developed in [23]. Traditional MC methods utilize known system dynamics to create a particle (ensemble) representation of time-varying state uncertainty. In turn, the ensemble is employed to compute desired statistics, e.g., various moments of the state. To begin, a finite set of initial realizations of the state are obtained by sampling the initial state-pdf, as indicated in (3), forming the initial ensemble. Each realization is forward propagated through the system dynamics to obtain the representation of the evolved state-pdf. In traditional MC, the size of the ensemble remains time invariant, whereby it is inevitable that the accuracy with which the propagated ensemble captures the true instantaneous state-pdf varies throughout the propagation process [38]. In other words, it is possible for the MC ensemble to be under- or over-sampled in varying degrees, which leads to lack of trust in the underlying simulations. Figure(2) presents a flowchart of an AMC simulation platform, which was first presented in [23]. The AMC platform improves traditional MC by creating an adaptive framework that allows for the addition and removal of particles from the MC ensemble, determined by the forecasting accuracy of application specific QoIs.

To enable adaptive ensemble control, the user is required to define appropriate QoIs. For each QoI, the user must also specify upper and lower error thresholds that represent performance bounds within which the simulation platform must operate: see the green shaded boxes in Fig.(2). The QoIs, along with their error thresholds serve as metrics to quantify the transient performance of the simulation (see Sec.(III-A)). The AMC platform defines adaptation rules in the form of an ensemble enhancer and ensemble thinner (see blocks labeled 4 and 5 in Fig.(2)) to maintain simulation performance within the user prescribed bounds at all times. When the performance of the current ensemble exceeds the upper error threshold, new optimal particles are introduced to the initial ensemble at t_0 and forward propagated to join the current ensemble at t_n [38]. The ensemble enhancer, (Block 4 in Fig.(2)) is continually executed until the measured QoI error falls back below the prescribed upper bound, $\mathcal{E}_t^{U^*}$.

On the other hand, depending on the variation of S_t (where $S_t = h \circ \Phi_t$ with $h = f(\mathbf{x})$ and Φ_t : the state-transition map), the measured accuracy of QoIs may exceed the requirements set by the user in terms of its lower error bound. If this happens, particles are selected for removal based on their current significance (weight) at time t_n via the ensemble thinner module (Block 5) and are halted for future propagation. Particles with lower weights are considered for removal with greater probabilities, where the weight of each particle is assumed to be the current value of the state-pdf evaluated at the particle's location. For the SDE given by (3), the time evolution of the state-pdf ($\mathcal{W}_t$) is governed by the corresponding SLE (4) that can be numerically integrated or solved with alternative approaches, detailed in [38]. The ensemble enrichment sub-modules (enhancer and thinner) together allow us to control the transient performance of the MC estimation [38]. Section.(III-A) details the theoretical basis for ensemble enhancements, and Sec.(III-B) details the current ensemble enhancer architecture. For an in-depth discussion on the other state-of-the-art AMC modules, please refer to [23].

A. QUANTITIES OF INTEREST

The AMC platform performs ensemble adaptations based on the difference between its measured forecasting accuracy and stipulated error bounds. QoIs are used to characterize the transient performance of MCS as it relates to $\mathcal{W}_t$. Each QoI is application specific and can be defined as anything from a simple state mean to the instantaneous heat flux on a vehicle, for example. In general, QoIs are defined as the expected value of a function of the state, $h(\mathbf{x}_t)$, where $\mathbf{x}_t$ is the current

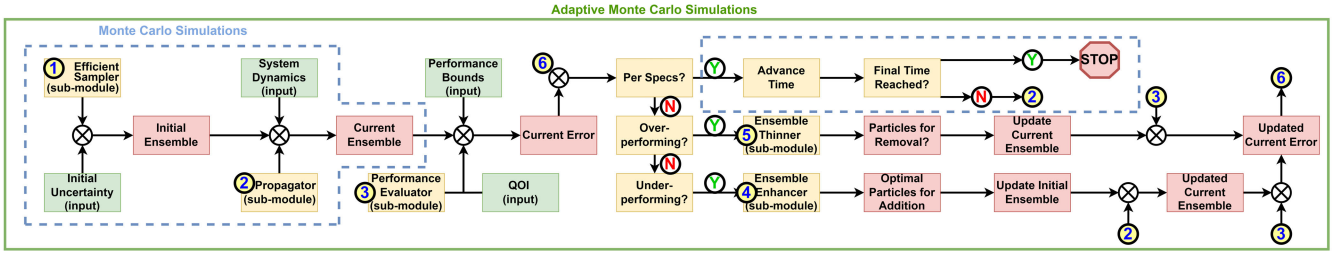

FIGURE 2. Adaptive Monte Carlo Simulations.

state with density function $\mathcal{W}(\mathbf{x}_t) \equiv \mathcal{W}_t$ [23]:

$$\underbrace{\overline{h}(\mathbf{x}_t) = \mathbb{E}_{\mathcal{W}_t}[h(\mathbf{x}_t)]}_{\text{Quantity of Interest: QoI}} = \int_{\Omega_t} h(\mathbf{x}_t)\mathcal{W}_t d\mathbf{x}_t \tag{8}$$

In the above expression, Ω_t is the state-space at time t. Prior to executing the AMC platform, its user must decide what application specific quantities ($\overline{h}(\mathbf{x_t})$) must be forecast within prescribed bounds, and, what those prescribed bounds need to be to achieve a trustworthy forecast. While this is a nontrivial task, the relationship between the QoI and the MC approximation error developed in [23] allows AMC to be implemented in a wide variety of scenarios (see Sec.(VI)). Continuing under the assumption that the state-transition map $\Phi_t(\mathbf{x}_0) = \mathbf{x_t}$ is injective and continuously differentiable, (8) can also be expressed in terms of the initial state-pdf:

$$\begin{aligned}\mathbb{E}_{\mathcal{W}_t}[h(\mathbf{x}_t)] &= \int_{\Omega_t} h(\mathbf{x}_t)\mathcal{W}_t d\mathbf{x}_t \\ &= \int_{\Omega_0} \underbrace{h[\Phi_t(\mathbf{x}_0)]}_{S_t(x_0)} \mathcal{W}_0 d\mathbf{x}_0 = \mathbb{E}_{\mathcal{W}_o}[S_t(\mathbf{x}_0)] \end{aligned} \tag{9}$$

where Ω_0 is the state-space at time t_0. $S_t(\cdot) \triangleq (h \circ \Phi_t)(\cdot) = h[\Phi_t(\cdot)]$ is an integrable composite function and $h(\cdot)$ is application dependent. For the complete derivation of (9), see [23]. QoIs form the basis of ensemble adaptations in the AMC platform through the so-called Koksma-Hlawka inequality given below in a modified form [23], [39]:

$$\begin{aligned}|\epsilon_n| = |\overline{f} - \tilde{f}_n| &= \left| \int_{\Omega_t} f(\mathbf{x}_t) d\mathbf{x}_t - \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x_t}^{\mathrm{i}}) \right| \\ &\leq \mathcal{D}(\{\mathbf{x}_t^i\}_{i=1}^n) V(f) \\ |\epsilon_n| = |\overline{h} - \tilde{h}_n| &= \left| \int_{\Omega_0} h[\Phi_t(\mathbf{x}_0)] d\mathbf{x}_0 - \frac{1}{n}\sum_{i=1}^{n} h[\Phi_t(\mathbf{x}_0^i)]) \right| \\ &\leq \mathcal{D}(\{\mathbf{x}_0^i\}_{i=1}^n) V(S_t) \end{aligned} \tag{10}$$

where $V(f)$ is the total variation of the function f(·), the samples $\mathbf{x_0}$ are drawn from a uniform distribution over Ω_0, $f \triangleq S_t = (h \circ \Phi_t)(\mathbf{x}_0) = h[\Phi_t(\mathbf{x}_0)]$, and $\tilde{h}_n$ is the sample-based approximation of $\overline{h}$ (with $\overline{h}(\mathbf{x}_t) = \mathbb{E}_{\mathcal{W}_t}[h(\mathbf{x}_t)]$ defined in (8)). That is, (10) develops the relationship between discrepancy with respect to the initial state-pdf (which is known) and the upper bound on the MC approximation error. The left hand side of (10) represents the departure of the AMC forecast from the true QoI value. As such, computationally efficient and easily estimated QoIs significantly reduce an unavoidable bottleneck in AMC and notably improve overall simulation run time. On the right hand side, $V(S_t)$ is the variation of the composite function S_t, and $\mathcal{D}(\{\mathbf{x}_0^i\}_{i=1}^n)$ denotes the discrepancy of the ensemble at the initial time t_0. The function $V(S_t)$ is not "controllable" and intractable for many high dimensional systems, but the discrepancy function, $\mathcal{D}(\{\mathbf{x}_0^i\}_{i=1}^n)$ is, since the initial state distribution is known. As a result, the *product* of terms on the right hand side of (10) can be controlled by performing an optimization of the discrepancy function.

B. ENSEMBLE ENHANCER

The redesign of the ensemble enhancer (module 4 in Fig.(2)) is the main focus of the present paper. The theoretical basis of particle addition in the AMC platform is the Koksma-Hlawka inequality (see Sec.(III-A)). That is, QoI forecasting error can be reduced by adding new particles that minimize ensemble discrepancy at the initial time. As mentioned before, this optimization problem is difficult, on account of high dimensionality and non-convexity of the discrepancy function. In the existing AMC platform, the actual optimization problem is never solved and new particles are introduced through an indirect reduction of discrepancy achieved by efficient sampling following the *non-collapsing* and *space-filling* properties. To introduce the ensemble enhancer sub-module within the current AMC platform, first assume that the current particle ensemble at time t can be represented by P_t^p with p particles. Let the corresponding initial ensemble at time t_0 be $P_{t_0}^p \sim \mathcal{U}[0,1)^N$ since a uniform ensemble can be transformed into any target state-pdf. Without loss of generality, assume the propagated mean is the tracked QoI which allows the current MC estimation error (standard deviation of the MC estimation error) to be estimated via bootstrapping and denoted by $\mathcal{E}_t^p = \mathcal{E}(P_t^p)$. Now, if the estimation error is greater than the user-prescribed threshold ($\mathcal{E}_t^P > \mathcal{E}_t^{U^*}$), the ensemble enhancer is activated and new particles are introduced until the MC accuracy falls back within the defined thresholds [23]. It should be noted that all particles are added at time t_0 to $P_{t_0}^p$ and then forward propagated to join the current ensemble at time t. For a complete description on the development of the ensemble enhancer please refer to [23]. The original ensemble enhancer within the AMC platform indirectly reduces discrepancy through a two-layered approach that

sequentially enforces criteria defined by the *space-filling* and *non-collapsing* properties of the ensemble to select particles for addition to $P^p_{t_0}$. A *space-filling* sample design leads to particles that fill out the domain of interest as homogeneously as possible [40]. Examples of some space-filling designs are Latin hypercubes, fractional designs, and orthogonal arrays [41], [42], [43]. There also exists several measures for quantifying this property, which include, but are not limited to, distance, entropy, and discrepancy. For the AMC platform, the discrepancy measure was chosen due to the relationship between the QoI (i.e., $\bar{h}(\mathbf{x}_t) = \mathbb{E}[h(\mathbf{x}_t)]$) and discrepancy given in Sec.(III-A) by the Koksma-Hlawka inequality. Discrepancy, $\mathcal{D}(\{\mathbf{x}^i_0\}^n_{i=1})$, can be numerically computed via the formula given for centered L_2 discrepancy in terms of an ensemble $P = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ with respect to a uniform distribution:

$$\mathcal{D}_{CL_2}(P) = \left\{ \sum_{u \neq 0} \int_{\mathbb{C}^u} \left| \frac{\#(P_u, J_{\mathbf{X}_u})}{n} - \mathrm{Vol}(J_{\mathbf{x}_u}) \right|^q d\mathbf{x} \right\}^{1/q} \tag{11}$$

where u is a nonempty subset of coordinate indices, $|u|$ is the number of elements in u, $\mathbb{C}^u$ is a unit cube in the $|u|$ dimensional space, $J_{\mathbf{X}}$ is an N-dimensional hypercuboidal volume, P_u is the projection of P to $\mathbb{C}^u$, and $J_{\mathbf{X}_u}$ is the projection of $J_{\mathbf{X}}$ onto $\mathbb{C}^u$ [26]. A more numerically tractable version of $\mathcal{D}(P)$ was derived by Hickernell and is given by (14) in Sec.(IV-A). The *non-collapsing* property assures that for every point $\mathbf{x}_i$, $(i = 1, \ldots, n)$, in a sampling design P, the coordinate of point $\mathbf{x}^j_i$, $(j = 1, \ldots, N)$, is strictly unique. That is, when a sampling design is projected onto a lower dimensional space, randomly chosen particles must not be superimposed [38]. This property can be defined as the minimum projected distance of points from one another [24]:

$$||P||_{-\infty} = \min_{\mathbf{x}_i, \mathbf{x}_j \in P} \min_{1 \leq k \leq N} |x^k_i - x^k_j| = \min_{\mathbf{x}_i, \mathbf{x}_j \in P} ||\mathbf{x}_i - \mathbf{x}_j||_{-\infty} \tag{12}$$

where $||\mathbf{x}||_{-\infty}$ is taken as the minus infinity norm. Equation(12) states each particle in the sampling design should have a unique coordinate along each dimension to avoid computational inefficiencies associated with essentially identical particles. The *non-collapsing* property is treated as the "gatekeeper" in the *admissible intervals* algorithm shown in Alg.(1). Looking again at (11), it becomes apparent why this criteria is important. Identical particles contribute in an identical way to (11). However, the ensemble size n increases regardless of if the particle is repeated, which also corresponds to an increase in computational load.

This two-layered routine is detailed in Alg.(1) with computational bottlenecks highlighted in red. The highlighted steps revolve around identifying and adequately exploring *admissible intervals*, which ensure the fulfillment of the *non-collapsing* property stated above. To do this, one must project the initial ensemble, P_{t_0}, onto each dimension and identify admissible intervals satisfying the projective distance

Algorithm 1 Ensemble Enhancer: Efficient Sampling

1: Estimate performance of the current ensemble P^n_t; that is, $\mathcal{E}^n_t$ via bootstrapping.
2: (Initialization): Project the existing initial ensemble $P^p_{t_0}$ onto each dimension of the state space.
3: (Identification): Identify admissible intervals along each dimension k that satisfy the projective distance threshold $\mathbb{Q}^k_q = \{\forall x^{k,c}_j \in \mathbb{Q}^k_q : \min_{x^k_i \in P^p_{t_0}} |x^k_i - x^{k,c}_j| \geq d_{\min}\}$ where x^k_i is the kth coordinate of $\mathbf{x}_i \in P^p_{t_0}$ and $d_{\min}$ is the minimum allowable projective distance.
4: (Intersection): Determine regions of intersection of admissible intervals along all dimensions ($\cap_{q,k}\mathbb{Q}^k_q$) and generate a candidate set in the regions of overlap $Q^{\mathcal{U}}_{t_0} \in \cap_{q,k}\mathbb{Q}^k_q$
5: (Cost): Compute and rank the cost of candidates within $Q^{\mathcal{U}}_{t_0}$ to determine the optimal candidate to be included in the initial ensemble.
6: Forward propagate to the next time instant t^+ when $\mathcal{E}^{L^*}_t \leq \mathcal{E}^n_t \leq \mathcal{E}^{U^*}_t$

threshold function:

$$\mathbb{Q}^k_q = \left\{ \forall x^{k,c}_j \in \mathbb{Q}^k_q : \min_{x^k_i \in P_{t_0}} \left| x^k_i - x^{k,c}_j \right| > d_{\min} \right\} \tag{13}$$

where x^k_i is the k^{th} coordinate of $\mathbf{X}_i \in P_{t_0}$, $k = 1, \ldots, N$, and $d_{\min}$ is the minimum allowable distance defined as $d_{\min} = \frac{\alpha}{p+1}$ [23]. α is a tuning parameter that controls the significance of the projective distance criterion [24]. Invoking (13) means that all candidates within an admissible interval $\mathbb{Q}^k_q$ are at a greater distance than $d_{\min}$ in each dimension from the current samples. It should be noted that candidates within admissible intervals do not necessarily satisfy (13) between *each other*. More so, the number of admissible intervals grows exponentially with the number of samples and one must assure each interval is adequately filled with candidate points. Appropriate filling is crucial in identifying the candidate corresponding to the global minimum of discrepancy to be included in the new ensemble and thereby in controlling the variability of the QoI seen in (10). As such, lines 3-4 of Alg.(1) increasingly become a bottleneck as the AMC ensemble grows in size, which is a direct consequence of complex system dynamics as well as strict performance requirements related to the QoI. It should be noted that even if admissible intervals were feasible as both n (ensemble size) and N (dimensional space) increase, this approach still circumvents the direct optimization of discrepancy (11) within the ensemble enhancer.

IV. RE-ENGINEERING AMC

Before redesigning the AMC platform, MATLAB®'s code profiler was utilized to determine potential bottlenecks within the individual modules slated for restructuring, arithmetic minimization, or parallelization. Profiling is a way to determine the time it takes to execute an application and

identify where MATLAB® spends most of its time to later be evaluated for possible performance improvements. The goal being the efficiency of algorithms not the coding language itself. Bottlenecks can be defined as areas within the AMC platform that cause congestion and starve subsequent sub-modules of data or where insufficient computational resources are available to handle the load which can also result in performance issues. Once a bottleneck is found, the corresponding component has to be fine-tuned in order to address the root cause. If insufficient computational resources are available, the CPU will reach its peak capacity early in the AMC execution timeline. Once at 100% capacity, the remaining sub-modules will have to queue until resources become available. Inadequate memory, network capacity, or heap size can also cause bottlenecks in a similar fashion. Another type of bottleneck is when a specific component or sub-module cannot provide data as quickly as needed by the rest of the application. That is, while the CPU is not at peak capacity a computation or component within AMC delays the execution of the rest of the platform due to data requirements from the bottlenecked component. In either case, a high resource consuming or non-distributable computation within AMC is likely the cause of any notable bottleneck and can be partially alleviated via arithmetic minimization and parallelization, which are discussed in Sec.(IV-A) and Sec.(V), respectively.

To conduct code profiling, a Lorenz-96 sample scenario was executed with the original AMC architecture and an upper accuracy bound of $\mathcal{E}_t^{U^*} = 0.1$ to determine the percentage of consumed computation time of major components within AMC. In Fig.(3), a flame graph shows a visual representation of the time spent on individual components of the AMC platform. MATLAB® functions are in gray whereas user-defined functions are in blue. The functions in the flame graph are shown in hierarchical order with parent functions appearing lower on the graph and child higher. The width of the bar labeled profile summary depicts the entire AMC platform runtime. The width of the individual bars above profile summary represent runtimes for individual modules and functions within MATLAB®. It should be noted that the profiler itself consumes some time, which is included in the results. For the ensemble enhancer routine, the computation time was further assessed to determine the exact line(s) of code contributing to the bottlenecks shown in Fig.(3). These bottlenecks can be tied directly back to the core of the *admissible intervals* routine within the ensemble enhancer and the discrepancy cost function. This analysis led to the arithmetic minimization of the discrepancy cost function and restructuring the ensemble enhancer to accommodate not only SA but also a novel parallel routine discussed in Sec.(IV-A), Sec.(IV-B), and Sec.(V-B), respectively.

A. DISCREPANCY COST FUNCTION

Discrepancy, (11), provides a means to control the variability of the quality of the estimation caused by the dependency

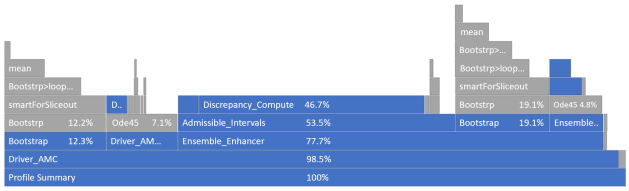

FIGURE 3. AMC Flame Graph.

on the variation function $V(S_t)$ seen in the Koksma-Hlawka inequality. It is at the heart of the ensemble enhancer and is shown to consume 46.7% of the run time depicted in Fig.(3). A more numerically tractable version of $\mathcal{D}(P)$ is (14) and was derived by Hickernell [44]:

$$\begin{aligned}\mathcal{D}^2_{CL_2}(P) &= \left(\frac{13}{12}\right)^N - \frac{2}{n}\sum_{k=1}^{n}\prod_{i=1}^{N}\left(1+\frac{1}{2}|x_{ki}-0.5| - \frac{1}{2}|x_{ki}-0.5|^2\right) \\ &\quad + \frac{1}{n^2}\cdot\sum_{k=1}^{n}\sum_{l=1}^{n}\prod_{i=1}^{N}\Big[1+\frac{1}{2}|x_{ki}-0.5| \\ &\qquad + \frac{1}{2}|x_{li}-0.5| - \frac{1}{2}|x_{ki}-x_{li}|\Big]\end{aligned} \tag{14}$$

where N is the dimension of the state space, and n is the current number of samples plus additional candidates. Examining (14), one should take special note of the dependency on the size of the state-space N as well as the ensemble size n especially under second term's double summation. This term is the mathematical representation of the bottleneck discussed below and will appear regardless of the ensemble enhancer executing the *admissible intervals* routine or SA introduced in Sec.(IV-B) due to numerous evaluations of (14).

The ensemble within the AMC platform is unique in that the current ensemble at t_0 is identical to the previous ensemble aside from the newly added optimal particle(s). This means that the outer summations in the first and second terms of (14) can be determined on a rolling basis, which eliminates the need for repeated computational effort. Even so, looking at the left hand side of Fig.(4) the computation time of (14) as implemented in the AMC platform shows a near linear increase in time as the ensemble grows and almost an increase of an order of magnitude for higher dimensional cases across a certain ensemble size threshold. That is, for the $6D$ Lorenz-96 case the computational cost for a single iteration of (14) jumps by an order of magnitude around 2.2×10^4 particles. This phenomenon can also be observed for a generic set of dynamic equations since (14) operates on $P^p_{t_0} \sim \mathcal{U}[0,1)^N$ as opposed to the initial target state-pdf ($\mathcal{W}_0$). This observation is important when either high levels of accuracy are required by the user or the propagation of system dynamics is complex as both can lead to large ensemble sizes and thereby an increase in computational cost. More so, discrepancy is at the core of the ensemble enhancement

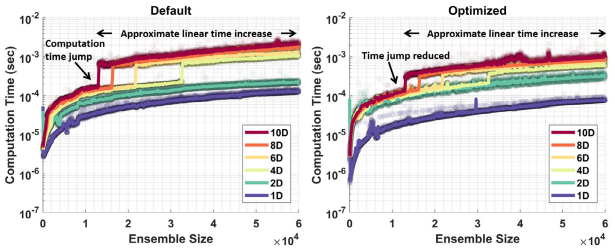

FIGURE 4. Equation(14) Clock Time in Increasing Dimensions.

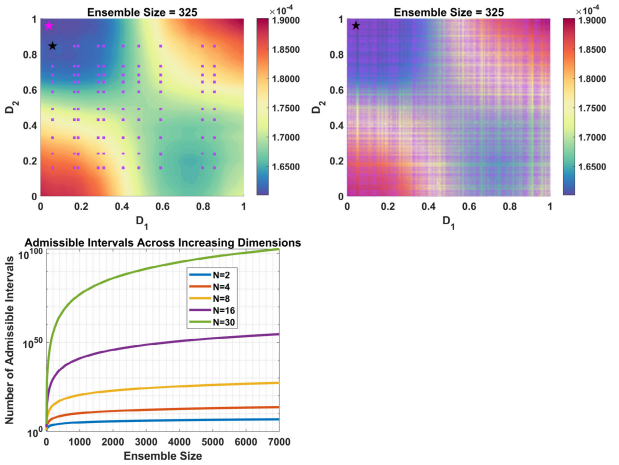

FIGURE 5. Increase of Admissible Intervals.

routine and is regularly computed during each *admissible intervals* or SA cycle, which highlights the effect of this increased cost.

The second portion of (14) involves minimizing the number of arithmetic operations as well as the examination of built-in MATLAB® mathematical functions such as absolute value, addition, and multiplication. The right hand side of Fig.(4) shows the difference between the pre- and post-arithmetic minimization of the discrepancy cost function that is valid for most computational languages. The updated algorithm minimizes the number of arithmetic operations by reordering the computation sequence as well as defining a common variable to be utilized in both portions of (14). It should also be noted that MATLAB® primarily operates along columns, meaning basic mathematical functions operating on large arrays are inherently quicker along the primary dimension. In general, one should take special note of the complexities involved in implementing standard mathematical libraries across any computational language to take advantage of efficient procedures and circumvent any notable pitfalls. In the context of the AMC platform and MATLAB®, the array passed to (14) should be an $N \times n$ matrix where N represents size of the state space, n the ensemble size, and $n \gg N$ in general.

B. ENSEMBLE ENHANCER: SA

The exponential growth of *admissible intervals* across increasing state-space dimensions (N) as well as ensemble size (n) is depicted in Fig.(5) and leaves the original ensemble enhancement routine impractical for large ensembles and high dimensional spaces. The figure pictorially represents (13) as $k = 1 \rightarrow N$ alongside larger ensembles. Examining the upper right side, one should note the difficulty in adequately filling out each interval with a candidate set as required for ranking in line 5 of Alg.(1). That is, for a sufficiently large ensemble size it is infeasible to check every available *admissible interval* to a identify the global minimum. A simple solution would be to rank the *admissible intervals* themselves based on the largest distance of each dimension and only place the candidate set within intervals that meet the specified minimum threshold. However, as can be seen from the upper left of Fig.(5) one can still incorrectly identify the global minimum (denoted in magenta) for a given ensemble when the number of intervals is limited in this manner. While different selection processes can be utilized in identifying the best *admissible intervals*, one can also look towards replacing this routine and solving the optimization problem defined by (15) directly:

$$\mathbf{x}_*^{p+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^N} \left[\mathcal{D}^2_{CL_2}(P_{t_0}^{(p+1)})\right] \tag{15}$$

where $\mathcal{D}^2_{CL_2}$ is (14) and $\mathbf{x}_*^{p+1}$ denotes the next optimal candidate to be included in $P_{t_0}^{p}$ thereby defining the $P_{t_0}^{(p+1)}$ ensemble.

To alleviate the computational burden of Alg.(1), Alg.(1) was replaced with Alg.(2) in [28], which utilizes a stochastic optimization routine known as SA to solve (15). SA is a class of gradient-free randomized algorithms that search for the global minimum of a cost function, in this case discrepancy for a given ensemble size n, by gradually reducing the magnitude of the random perturbations with respect to the current state. Algorithm(2) details the basics of SA whereby discrepancy (14) is directly optimized within the ensemble enhancer for every particle addition. For current purposes, we optimized (via brute force) the cooling rate to be set at $\gamma = 0.3$ and the standard deviation of the proposal density at $\sigma = 0.1$. Figure(6) illustrates the time history of the cost function (discrepancy (14)) for a randomly generated ensemble alongside SA's exploration history. The darker diamonds on the right side of Fig.(6) reflect a larger number of iterations spent idled in position. It should be noted that SA does not identically match the true global minimum (6.146×10^{-5}) mainly due to SA's convergence properties associated with specific cooling schedules [45], [46]. While important to recognize, the essence and success of SA is in its ability to accept an increase in the cost function (shown in Fig.(6)) with probability < 1 thereby avoiding entrapment in local troughs, not necessarily in its ability to quickly converge to the global minimum with absolute certainty [47]. To balance convergence and computational efficiency, the parameters $\gamma = 0.3$ and $\sigma = 0.1$ were tuned for adequate exploration of the domain $[0, 1)^N$ at higher temperatures ($T_{\max} = 35$). Since k in line 9 of Alg.(2) is also a tuning parameter associated with the acceptance criteria, it was chosen to normalize δ as $|\delta| < 1 \times 10^{-5}$ for sufficiently large ensembles. Without

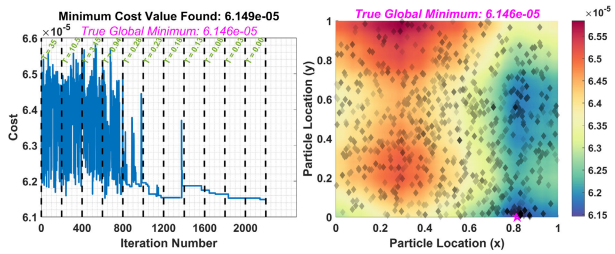

FIGURE 6. SA Cost and Exploration History.

normalization, $\exp(-\delta/kT) \approx 1$ as $T \to 0$ meaning the probability of acceptance is ≈ 1 irrespective of temperature. Requiring normalization, setting $T_{\max} = 35$, $\gamma = 0.3$, and $\sigma = 0.1$ results in sufficient domain exploration in the early stages of Alg.(2), which can be seen in Fig.(6). The cycles per temperature were also optimized to be $n_{\text{cyc}} = 200$ and the cooling schedule was slightly modified as follows to help with computational efficiency:

$$T = \begin{cases} \gamma T, & \text{if } T > 0.5 \\ \max(T - 0.05,\ T_{\min}), & \text{otherwise} \end{cases} \tag{16}$$

where $T_{\min} = 0$ (i.e. only *cost reductions* are accepted with certainty during the last cycle of Alg.(2)).

The authors note that this combination of tuning parameters is atypical for traditional simulated annealing and is more in line with a philosophy known as simulated quenching. In particular, the quenching schedule and tuned number of iterations *per particle* in Alg.(2) may appear quick as compared to traditional annealing. However, an avenue of compensation is provided by reformulating Alg.(2) for parallelization and leveraging intermediate solution exchange, which is detailed later in Sec.(V). Broadly speaking, quenching potentially trades computational efficiency at the cost of the necessary and sufficient conditions required for convergence to the global minimum [48]. With proper tuning, SQ has still seen success in a wide array of practical applications [49].

As is, SA has no concept of the *non-collapsing* property defined previously in the two-layered approach while it searches for the candidate corresponding to the global minimum of the discrepancy cost function. That is, SA may violate the *non-collapsing* property when choosing optimal candidates in terms of discrepancy alone. However, SA represents significant computational savings and reliability as compared to *admissible intervals* since it is not required to fill all areas of interest with candidates. SA simply employs a randomized search algorithm in relation to its current chain position. This alleviates the need for ranking individual candidates to determine the global minimum done by Alg.(1). The computational time savings provided by Alg.(2) are also improved upon through the arithmetic minimization of the discrepancy cost function detailed Sec.(IV-A). The new ensemble enhancer (see Fig.(2)) is further restructured in Sec.(V) via a novel parallelization algorithm utilized to solve a sequence of optimization problems and continually reduce the computational burden of AMC. That is, Sec.(V) details

Algorithm 2 Ensemble Enhancer: SA

1: Estimate performance of the current ensemble P_t^n; that is, $\mathcal{E}_t^n$ via bootstrapping.
2: (Initialization): Set an initial temperature $T = T_{\max}$ and a starting point for the search $x_0 = x_{\text{curr}} \in \Omega$. Determine $L(x_{\text{curr}})$
3: (Transition Proposal): Sample candidate $x_{\text{new}} \in \Omega$ from a proposal density $q(x)$, e.g. $q(x) = \mathcal{N}(x_{\text{curr}}, \sigma)$. Determine $L(x_{\text{new}})$ and compute the change in cost $\delta = L(x_{\text{new}}) - L(x_{\text{curr}})$
4: (Acceptance): There are two cases
5: **if** $\delta < 0$ [*Cost Reduction*] **then**
6: Accept the candidate x_{new}, i.e. $x_{\text{curr}} \leftarrow x_{\text{new}}$ and $L(x_{\text{curr}}) \leftarrow L(x_{\text{new}})$
7: **else if** $\delta \geq 0$ [*Cost Increase*] **then**
8: Draw $u \sim \mathcal{U}[0, 1]$
9: **if** $u \leq \exp(-\delta/kT)$ **then**
10: Accept the candidate x_{new}, i.e. $x_{\text{curr}} \leftarrow x_{\text{new}}$ and $L(x_{\text{curr}}) \leftarrow L(x_{\text{new}})$. (**Metropolis Acceptance**)
11: **else**
12: x_{curr} does not change
13: **end if**
14: **end if**
15: (End Chain for Current T): Continue to evolve changes (steps (2-11)) until resources (cylces) for the current temperature is exhausted.
16: (Cooling): Reduce T according to the prescribed cooling schedule, e.g $T \leftarrow \gamma T$ and return to chain evolution (steps 2-11).
17: (End): Terminate optimization if the temperature is reduced below a threshold value ($T < T_{\min}$) or if the max computational limit is reached. Return $x^* = x_{\text{curr}}$.
18: Forward propagate to the next time instant t^+ when $\mathcal{E}_t^{L^*} \leq \mathcal{E}_t^n \leq \mathcal{E}_t^{U^*}$

the exploitation of SA's cost function history to include m additional particles in parallel under the same clock time as sequential SA utilizing *near optimal* estimates of candidate particles up to the $U_0 + (i-1)^{\text{th}}$ ensemble.

V. PARALLELIZATION

Parallelization can be defined as the process of taking a serial code application that runs on a single CPU and spreading the workload across multiple cores towards the goal of significantly reducing simulation time. Transitioning the AMC platform from serial to parallel requires non-trivial code rewriting detailed in Sec.(IV-A) and Sec.(V-B). When designing an algorithm in parallel, it is useful to consider various approaches in order to analyze competing performance gains. Broadly speaking, parallel programming should focus on the following properties: performance, productivity, and portability. That is, in parallel programming it should be predictable to achieve good performance, scalable, maintainable, efficient, and functional across multiple platforms [50]. While the minimization of the total amount of computational

work is an inherent goal of parallelization, communication or access to memory may also frequently constrain performance. In the context of the AMC platform, communication between CPU cores is an important consideration in the design of parallel SA detailed in Sec.(V-B). One should also take special note of the longest chain of tasks that must be performed sequentially within the AMC platform, which leads to an additional constraint on parallel performance [50]. Examining Fig.(3), the longest chain of serial tasks in AMC is contained in the ensemble enhancer routine in which (14) is minimized via SA to determine globally optimal candidates. That is, each CPU core evaluates (14) a predetermined number of times to search for the global minimum of the cost function. As detailed in Sec.(IV-B), SA depends on the current cost function realization and temperature values, meaning evaluations of (14) per particle addition are not the main focus of parallelization here. Instead, multiple instances of the restructured ensemble enhancer routine are parallelized to add m particles concurrently, the details of which are in Sec.(V-B).

For AMC, parallelization focused on data parallelism and functional decomposition. Data parallelism is any kind of parallelism that grows with the data set whereas functional decomposition runs different program functions in parallel. Section (V-B) details the specifics of the novel parallel SA routine and can be classified as the main focus for functional decomposition in AMC. At best, functional decomposition improves performance by a constant factor depending on the number of subfunctions, similar execution time, and overhead [51]. However, functional decomposition can provide additional parallelism to meet performance requirements given easily accessible computational resources. AMC also employs data parallelism to scale past a constant factor improvement when performing accuracy estimation. It should be noted that the serial performance of modules in the AMC platform, like those discussed in Sec.(IV-A), cannot be neglected as inefficient implementations will remain in parallel. That is, even within a parallel thread, computations and tasks are ultimately carried out by serial code [51].

A. STATE OF THE ART: PARALLEL SA

As described in [28], SA offers an alternative approach to the original architecture of the ensemble enhancer within AMC. SA has been utilized to study the traveling salesman problem, in circuit design, the design of decision trees, in data analysis, imaging and neural networks, as well as in areas of biology, physics, finance, and the military, among others [52], [53], [54], [55], [56], [57], [58], [59], [60], [61]. As SA is inherently sequential, significant research has been conducted to not only increase its serial efficiency, but also to parallelize the algorithm while maintaining convergence properties or, at the very least, induce minimal errors in the algorithm. Enhancements on the sequential side include: quenching, mean-field annealing, simulated tempering, fast annealing, generalized SA, threshold SA, thermodynamic SA, and information guided SA [47], [61], [62], [63], [64], [65], [66], [67], [68]. Simulated quenching (SQ) can

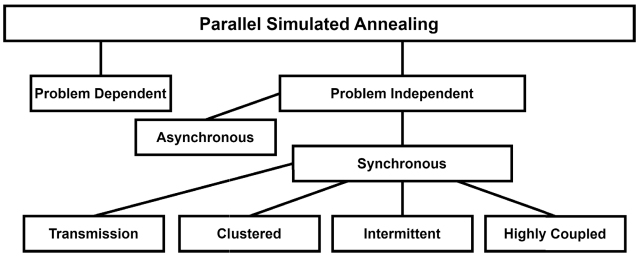

FIGURE 7. Parallel SA Taxonomy.

be thought of as a greedy algorithm trading computational efficiency at the detriment of SA convergence properties. However, with proper tuning, SQ has seen success in a wide array of applications [49]. In [62], mean-field annealing relies on the mean values of variables being a good approximation to the optimal stochastic state for quasi-quadratic energy functions. Simulated tempering attempts to maintain the equilibrium of a system while seeking alternative minima thereby lowering the effective cost [63]. Szu et al. [64] developed fast SA by leveraging Cauchy distributions which led to faster exploration and convergence. Hoffmann et al. [67] prove that threshold accepting is the best possible strategy for SA for a wide range of objective functions. Thermodynamic SA develops an optimal operating strategy for minimizing entropy [68]. In [47], information guided SA leverages information gathered during the randomized exploration stage to use as feedback for driving the optimization procedure. While the above developments are noteworthy in the field of SA, none attempt to leverage parallel processing to further increase the efficiency of SA.

Figure(7) categorizes parallel SA into several different categories and sub-categories namely: problem dependent/independent, asynchronous and synchronous, as well as clustered, intermittent, and highly coupled [69]. Problem dependent parallelization can be thought of as partitioning the problem across several processors and communicating only when dependencies necessitate it (e.g., near subdomain boundaries). Problem independent parallelization can take on various different meanings as seen in Fig.(7). In asynchronous, independent SA chains are instantiated with differing initial conditions, executed for a predetermined number of runs, and ranked to then report the best solution. For synchronous SA, transmission, clustered, intermittent, and highly coupled differ only in the frequency of solution exchange and worker responsibilities thereby contributing to a higher or lower amount of communication overhead. Additionally, literature such as [70] classifies parallel SA algorithms as either Single Markov Chain SA (SMCSA) or Multiple Markov Chains SA (MMCSA). SMCSA can be thought of as a version of highly coupled or transmission SA whereas MMCSA would be classified under asynchronous, intermittent, or clustered SA. It should be noted that hybrid parallel algorithms have also been developed that utilized both SMCSA and MMCSA throughout the different stages of SA. In fact, [71] details a solution methodology that exploits

both SMCSA and MMCSA dependent on the acceptance ratio of generated candidate solutions. An example of asynchronous SA is the implementation of MaxMove perturbations described in [69] to report the best solution amongst all workers. Onbaşoğlu et al. [69] also develop versions of SA for the additional categories listed in Fig.(7) and perform extensive benchmarking against 106 test functions for consistency and comparison against other state-of-the-art SA algorithms [72].

More recent developments in parallel SA include: simplex method, adaptive resampling interval, parallel stretched SA, greedy, and coupled [73], [74], [75], [76], [77]. The simplex method in [73] is a hybrid algorithm where SA is applied to each simplex thereby producing tentative solutions. The adaptive resampling strategy allows each processor to independently choose (probabilistically) a target state during resampling and subsequently carry on annealing [74]. In [75], parallel stretched SA allows for increased resolution of the search domains while bounding search time. The greedy algorithm discussed in [76] implements a memorization technique to conduct parallel local searches with multiple local search instances sharing intermediate solutions. New asynchronous/synchronous parallel coupled SA algorithms are also proposed in [77] where each parallel thread is responsible for assessing its cost before entering a barrier stage (synchronous) or entering a critical section asynchronously.

B. PARALLEL SA FOR DEPENDENT COSTS

While the strategies listed in Sec.(V-A) can be leveraged for the optimization of a *single* instance of discrepancy, the AMC platform requires the global optimal solution to (14) for each additional particle to be included in the ensemble. Therefore in AMC, the parallelization of (14) (discrepancy) for increasing ensemble sizes is chosen as opposed to parallelizing lines 3 − 13 of Alg.(2) with the notion that the strategies developed above can be incorporated alongside the technique here as the AMC platform continues to evolve. In terms of classification, the parallel SA algorithm developed here should be considered Single Markov Chain Quenching due to the modified cooling schedule and each processor being responsible for a *single* instance of discrepancy. Restated, a sequence of optimization problems will be parallelized across m cores by leveraging SA since (14) is uniquely defined for every additional particle incorporated into the ensemble. Each CPU is currently responsible for the addition of *one* new particle (dependent on all parent CPUs) with batch addition per core planned for future work. That is, each CPU is tasked with solving consecutive cost functions defined by (17) (and visualized in Fig.(8)):

$$\mathbf{x}_*^{p+m} = \arg\min_{\mathbf{x}\in\mathbb{R}^N} \left[\mathcal{D}^2_{CL_2}(P_{t_0}^{(p+m)}) \right] \tag{17}$$

where $m = 1, \ldots, i$, i the number of available CPUs, $D^2_{CL_2}$ is (14), and $P_{t_0}^{(p+m)}$ defines an optimal ensemble of $p + m$ particles dependent on all previous solutions up to $m - 1$. One should note the difference between (15) in Sec.(IV-B) and (17) is in the $p + 1$ and $p + m$ superscripts highlighting the requirement of solving successive and unique cost functions for every particle added sequentially or in parallel.

In parallel, each child CPU is able to leverage the exploration history of its parents' to determine the optimal candidate of the $U_0 + i^{\text{th}}$ ensemble. Throughout the execution timeline of parallel SA, children CPUs utilize temporary *near optimal* estimates of candidate particles up to the $U_0 + (i-1)^{\text{th}}$ ensemble, which are continually updated at the top of each temperature cycle thereby perturbing all children cost functions. The higher the number of parallel CPUs, the higher number of temporary values children CPUs are utilizing to optimize their *own* cost function. While the architecture of Fig.(8) is straightforward, a few integration items must be addressed, namely: (1) SA's stochastic nature, (2) resychronization of parallel CPUs, (3) initialization of SA, and (4) adequate exploration of the domain. Firstly, since SA is stochastic in nature the temperature cycles of each parent and child CPU may not exhibit the same computational effort even if they are beginning from identical temperatures. That is, CPU_1 and CPU_2 may complete the cycle corresponding to $T = 35$ (for example) asynchronously. This can be traced back to line 2 as well as lines 4-11 in Alg.(2). The AMC platform implements a resynchronization step at the end of each temperature cycle to avoid extreme delays of the SA executation timeline pertaining to children CPUs. In essence, AMC forces all CPUs to wait for the completion of a specific temperature cycle before proceeding to line 13 in Alg.(2). This approach was chosen to avoid scenarios in which children CPU timelines were delayed by a significant amount such that SA was nearly operating in a sequential fashion with *stale* parent updates as opposed to parallel. Another design choice the AMC platform makes is the serial computation of the first temperature cycle for all CPUs. As stated in [28] and depicted in Fig.(6), the magnitude of the cost function perturbation inherited by the children CPUs is designed to settle as the execution timeline progresses. By computing the first temperature cycle in serial, the child CPUs are able to adequately explore the domain space of their new cost functions at the highest temperature given the current best estimate from their parents. Alternative approaches to this include: reheating the child temperature based upon the corresponding parent temperature of the inherited candidate or increasing the temperature cycle iteration number in a similar fashion. Regardless of the approach, the goal remains to explore the domain space of the perturbed cost function, especially when the perturbations are large. When SA cools, exploration of the entire domain becomes less important as the cost function related to T_i and T_{i-1} is nearly identical. Even though parallel SA alleviates a significant computational burden within the ensemble enhancer, data communication and resynchronization of each CPU in parallel does introduce noteworthy clock time overhead in comparison to serial computations, which should be considered [28], [78].

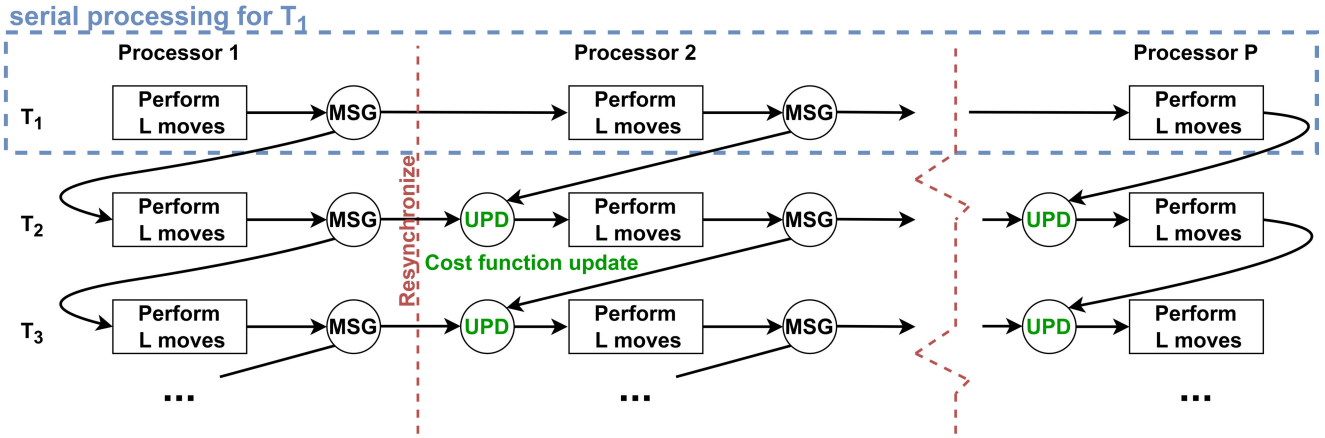

FIGURE 8. Parallel SA for Dependent Cost Functions.

C. CHALLENGES OF PARALLEL SA

The architecture of parallelization in AMC depends on multiple processors communicating via shared memory in MATLAB®. It should be noted that parallelization introduces additional overhead to launch and synchronize tasks, thereby increasing both the work and span of the simulation albeit still notably quicker than serial SA and substantially faster than *admissible intervals*. As such, there is a tension between decomposing tasks for load balancing while still keeping them large enough such that the cost of synchronization is negligible and arithmetic intensity is maximized [51]. In AMC, the main sub-module that requires balancing in parallel is SA. The core of parallel SA discussed in Sec.(V-B) is the optimization of (14), namely, the discrepancy cost function. As shown in Fig.(4), the computation of discrepancy is dependent on the state dimension (N) as well as the ensemble size (n), even after arithmetic minimization. To implement parallel SA, the immediate parent CPU ensemble needs to be transmitted to the child CPU, which is depicted in Fig.(8). As the state dimension and ensemble size grow, this becomes a communication bottleneck in shared memory. That is, passing an ensemble of size $n = 10^4$ versus $n = 10^2$ particles requires significantly more computation time. This is compounded by the fact that discrepancy is minimized at each pass of the ensemble enhancement routine. To circumvent this, the current true MC particle ensemble lives as a static variable on each core with local ensemble updates being performed during the SA execution timeline shown in Fig.(8).

Global ensemble updates occur after each group of CPUs have completed their respective SA routines. If a large number of CPU cores are available to the AMC platform, one should also take note of the potential degradation in the quality of the ensemble, which is quantified via (14). This phenomenon can be traced to the difference between local and global updates of the particle ensemble. During parallel SA, child CPUs depend upon *near optimal* estimates of the $U_0 + (i-1)^{\text{th}}$ ensemble as each iteration of parallel SA executes synchronously. The child CPU executes its optimization routine with the parent's *near optimal* estimate until an update is received. When an update is received, the child's cost function changes and the routine continues. It should be noted that these updates are the local updates discussed above. When a batch of CPUs complete their execution timelines, a global update is performed on the particle ensemble. This is equivalent to updating the particle ensemble with each *optimal* particle as determined by SA. The difference between running 100 CPUs in parallel versus 25 CPUs across 4 iterations is in the global updates of the particle ensemble as well as the computation time. The former would be expected to execute quicker whereas the latter may better maintain the quality of the overall ensemble. Depending on the computation time and internal AMC performance requirements, the tradeoff between ensemble quality and simulation time can be tuned on a case-by-base basis. It should also be noted that an order of magnitude change in discrepancy does not correspond to a significant difference in ensemble sizes for the use cases analyzed in [28] and [78]. That is, even though parallel SA may slightly degrade overall ensemble quality as measured by discrepancy, the accuracy of the QoI is largely unaffected and, correspondingly, an increase in the ensemble size has not been observed.

D. PARALLEL SA EFFECTS ON DISCREPANCY

To further illustrate the discussion above, an example entry, descent, and landing scenario utilizing the dynamics defined in Sec.(II-A) and an upper accuracy bound of $\mathcal{E}_t^{U^*} = 0.2$ W/cm^2 on the QoI (heat flux to be defined in Sec.(VI-A)) was executed in parallel for $1, 4, 8, 18$, and 52 CPU cores as well as a basic random sampling Monte Carlo (BRS-MC) simulation for comparison. The results pertaining to the degradation of the ensemble's discrepancy are highlighted in Fig.(9). Each core in parallel SA is required to leverage *near optimal* estimates of particles up to the $U_0 + (i-1)^{\text{th}}$ ensemble. Again, the higher the number of parallel CPUs, the higher number of temporary values children CPUs are utilizing to optimize their *own* cost function. More so, child

cost function perturbations occur as updates are passed down from their parent(s). The more parents a child has, the higher the chance a perturbation occurs (see Fig.(8)). Since the temporary values passed from parent to child are *nearly optimal*, parallel computing is expected to slightly degrade the overall ensemble's discrepancy with respect to sequential annealing, as can be observed in Fig.(9). As the number of cores increase from 1 to 52, one can note the degradation (increase) in discrepancy for a given ensemble size. However, this degradation is always below that of the BRS-MC simulation that utilized 153, 500 particles to meet an accuracy threshold of $\mathcal{E}_t^{U^*}$ at all times. The ending ensemble sizes for parallel SA are as follows: 148895, 149451, 147615, 150261, and 149047 particles which correspond to 1, 4, 8, 18, and 52 parallel CPUs and are notably below that of the number of particles required by BRS-MC.

Examining Fig.(9), one can see the discrepancy of parallel SA for CPUs 4 and 8 is distinctly closer to sequential annealing and less noisy than the others, which further reinforces the trade-off between ensemble quality and computation time. This also supports the analysis done in [28] and [78] that stated an order of magnitude change in discrepancy did not correspond to a significant difference in ensemble sizes for the cases analyzed. It is interesting to note that parallel SA with 8 CPUs meets the required accuracy threshold with the fewest number of particles (and clock time as will be reported in Sec.(VI)) in conjunction with minimal discrepancy degradation. At face value, one would expect the architecture with the lowest discrepancy for a given ensemble size to meet the accuracy threshold quickest. However, adaptations within AMC are initiated when the QoI, which in this example scenario is a function of the state, exceeds the upper threshold *not* when discrepancy exceeds a certain bound. The phenomenon observed is associated with the relationship embedded between $\mathcal{D}(\{\mathbf{X}_0^i\}_{i=1}^n)$ and $V(S_t)$ as shown in the Koksma-Hlawka inequality (Sec.(III-A)). That is, there are scenarios in which an ensemble with a slightly degraded discrepancy will meet $\mathcal{E}_t^{U^*}$ with fewer particles than the ensemble with the optimal under the assumption that both are below the would be enforced upper bound of the Koksma-Hlawka inequality. While determining the exact upper bound given by the Koksma-Hlawka inequality is worthwhile and can be computed for simple one dimensional systems, $V(S_t)$ is typically application dependent and complex or intractable for higher dimensions and therefore out of scope for the current article. In general, proposed parallel architectures should maintain closeness to the optimal discrepancy (sequential SA), which will ultimately lead to similar ensemble sizes that are consistently below that of BRS-MC. This result can be further observed in Sec.(VI) for numerous scenarios.

VI. RESULTS

This section details the results of entry, descent, and landing, high dimensional Lorenz-96, and Lotka-Volterra use cases as they relate to the re-engineered AMC platform. The results in the succeeding sections were generated using a Dell Precision 3240 Compact Workstation with an Intel(R) Xeon(R) W-1290 CPU @ 3.20GHz and 32.0GB of RAM. All code minimization, optimization, and parallelization results discussed above were implemented within the AMC platform. That being said, an entry, descent, and landing use case is first introduced in Sec.(VI-A) and analyzed under the context of overall simulation time and ensemble quality. This is then followed by high dimensional Lorenz-96 and Lotka-Volterra models to examine AMC performance as the ensemble sizes and dimension of the state-space increase.

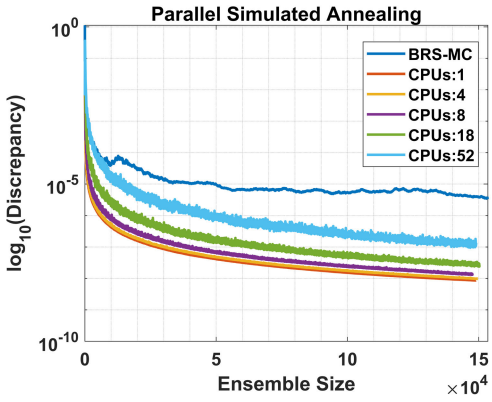

FIGURE 9. Parallel SA Effects on Discrepancy.

A. ENTRY, DESCENT, AND LANDING

Consider Vinh's equations introduced in Sec.(II-A) (5), detailing simplified hypersonic entry, descent, and landing (EDL) dynamics [27]. The dynamics are assumed to be purely longitudinal for a nonrotating spherical planet with zero bank-angle flight. The central object is assumed to be Mars, where $R_0 = 3.397 \times 10^6$ m and $\mu = 4.282837 \times 10^{13}$ m^3/s^2. An exponential model described as follows is used to determine the density of the Martian atmosphere:

$$\rho = \rho_0 \exp\left(\frac{h_2 - hR_0}{h_1}\right) \tag{18}$$

where $\rho_0 = 0.0019$ kg/m^3, $h_2 = 9 \times 10^4$ m, and $h_1 = 9.8 \times 10^3$ m. The following initial conditions are assumed [79]: $h_0 = 8 \times 10^4$ m, $V_0 = 3.5$ kg/m, and $\gamma_0 = -2°$. The initial states are assumed to have Gaussian uncertainty about their respective nominals with 10% variance for each state variable. The length and time variables were also nondimensionalized utilizing Mars' mean equatorial radius as the reference length (R_0) and the period of circular orbit at R_0 ($t_0 = R_0/v_c$ where $v_c = \sqrt{\mu/R_o}$).

In hypersonic EDL, accurate prediction of the heating sustained by a vehicle during flight can aid in the design of its thermal protection system [80]. As such, the instantaneous heat flux at a point on the entry vehicle is of interest and can be defined as:

$$\dot{Q}(h, V) = \frac{1}{4} C_f v_c^3 \rho V^3 S = \frac{C_f \rho_0 v_c^3 S}{4} \exp\left(\frac{h_2 - hR_0}{h_1}\right) V^3 \tag{19}$$

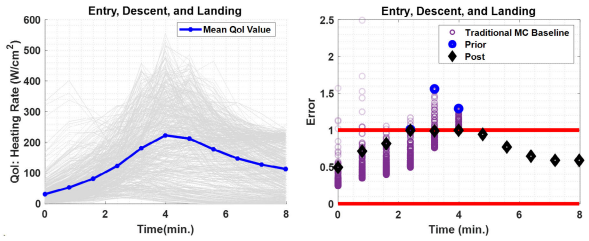

FIGURE 10. Entry, Descent, and Landing: QoI.

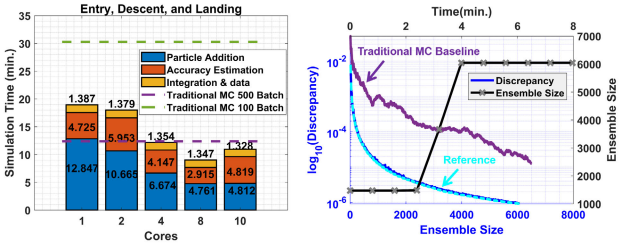

FIGURE 11. Entry, Descent, and Landing: Simulation Time, Ensemble Size, and Discrepancy.

where $S = 5\,\text{m}^2$ and $C_f = 1.0$ to simulate a spacecraft similar to the Mars Science Laboratory mission [80]. Therefore, the expected heating rate during propagation can be defined as the QoI within the AMC platform as:

$$\underbrace{\bar{h}(\mathbf{x}_t) = \mathbb{E}_{\mathcal{W}_t}[\dot{Q}(h, V)]}_{\text{Quantity of Interest: QoI}} = \frac{C_f \rho_0 v_c^3 S}{4} \int_{\Omega_t} \exp\left(\frac{h_2 - hR_0}{h_1}\right) V^3 \mathcal{W}_t \, \mathrm{d}\mathbf{x}_t \qquad (20)$$

The lower and upper estimation error bounds in approximating the QoI were set to be $\mathcal{E}_t^{L^*} = 0\ \text{W/cm}^2$ (ensemble thinner turned off) and $\mathcal{E}_t^{U^*} = 1\ \text{W/cm}^2$. The nondimensional forward propagation time is taken to be $\tilde{t}_0 = 0$ to $\tilde{t}_f = 0.5$, corresponding to approximately 8 minutes of real time [23].

The left hand side of Fig.(10) depicts the time evolution of (20) as the spacecraft descends through the Martian atmosphere. Underlayed are specific particle instances of the QoI corresponding to the associated uncertainty. During the middle stages of the flight, the spacecraft experiences the most severe heating (beginning around $t = 2 - 4$ min.) and thereby triggers the ensemble adaptations shown on the right hand side. That is, QoI estimation error is shown using black diamonds when the measured error is within the prescribed bounds and blue circles when ensemble adaptations are required. The purple error curve on the right hand side of Fig.(10) shows the evolution of a BRS-MC simulation implemented to meet the prescribed accuracy bounds of the AMC platform. To do this, batches of 500 particles were added to the existing MC ensemble, propagated forward in time, and analyzed in the performance evaluator (module 3 in Fig.(2)) of the AMC platform. If the performance bounds were violated, the parallel BRS-MC simulation was reinitialized with an additional set of 500 particles until the performance thresholds were met at all times. This leads to the parallel BRS-MC ensemble outperforming the AMC platform's error in the early stages due to a larger overall ensemble prior to the adaptations performed by AMC. However, parallel BRS-MC comes with no guarantees since there is no true performance control in this approach and no mechanisms available to know a-priori how many particles would prove "sufficient" for the entire simulation [23]. The computational cost of performance guarantees comes in the form of executing the ensemble enhancer (module 4 in Fig.(2)) with associated simulation times shown in Fig.(11) alongside the parallel BRS-MC simulation.

As described in Fig.(11), the computation time breakdown for the AMC is shown on the left hand side along with baselines for the parallel BRS-MC simulation with 100 and 500 batch particle additions utilizing the optimal number of parallel cores for components outside of the ensemble enhancer (ie: 8 cores here pertaining to the minimum clock time of modules 2 and 3 in Fig.(2)). It should be noted that a parallel BRS-MC simulation with the exact number of particles needed as determined by AMC is not shown as this number is not known a-priori. The author notes that including this simulation would by-pass checking $\mathcal{E}_t^{U^*}$ and the need for the ensemble enhancer as AMC would had to have already been utilized prior in determining the ensemble size with sufficiently accurate results found as well. A simulation with a significantly large number of particles (several orders of magnitude above those shown in Fig.(11)) can be executed. However, the quality of the ensemble will follow a similar trend to that of the traditional MC baseline shown in Fig.(11) with the computational burden placed solely on the propagator (module 3 in Fig.(2)). That is, the quality of a large traditional MC ensemble will be significantly degraded as compared to AMC. This means a larger ensemble at more distant future time steps will eventually be required to meet the accuracy bounds that would have been prescribed by AMC. It should be noted that the difficulty in accurately estimating a prescribed QoI also plays a role in determining the ensemble size (refer back to the Koksma-Klawka inequality). As such, comparing a parallel BRS-MC simulation with a given ensemble size determined a-priori by AMC or a substantially larger ensemble would simply be comparing the computational trade-off of propagating a large ensemble through the same dynamic system AMC is utilizing with that of other modules within the platform. There is little insight to gain here as the propagator (module 3 in Fig.(2)) is fixed for both AMC and BRS-MC.

For the parallel BRS-MC simulation with 100 and 500 batch particle additions, computation time varies from a maximum of 30.28 min. to a minimum of 12.38 min. Without the performance guarantee provided by the AMC platform, the parallel BRS-MC simulation was required to restart at each violation of $\mathcal{E}_t^{U^*}$, which significantly slowed down the overall simulation time. As the number of parallel CPU cores is swept from $1 - 10$ cores (with 1 CPU denoting sequential SA), the AMC simulation time

reaches a minimum of 9.02 min. at 8 cores with the largest fraction of time being consumed by the ensemble enhancer routine. Correspondingly, the parallel AMC platform with 8 CPU cores outperforms the large batch size parallel BRS-MC simulation, meaning the performance guarantee CPU "cost" is negligible when compared to parallel BRS-MC. The right hand side of Fig.(11) shows the evolution of (14) (discrepancy) of the AMC platform's ensemble in dark blue and the reference sequential SA (cyan) alongside the would-be discrepancy of the parallel BRS-MC ensemble. As discrepancy is a measure of non-uniformity for the sample set, one can conclude the AMC platform's ensemble better represents the uncertainty associated for the given EDL dynamics (5) and for a given ensemble size even though each ensemble meets the required accuracy thresholds. The right hand side of Fig.(11) also depicts the time evolution of the ensemble size related to the AMC platform. As can be seen, the ensemble increases from an initial size of 1475 particles to 1485 at 2.39 min., 3665 at 3.19 min., and finally to 6045 particles for the remainder of the simulation. This ensemble size increase directly corresponds to violating $\mathcal{E}_t^{U^*} = 1$ W/cm^2 at 2.39 min., 3.19 min., and 3.98 min. as shown in Fig.(10).

B. LORENZ-96 MODEL

As defined in Sec.(II-B), the Lorenz-96 system contains K state-space variables $(x_1, \ldots, x_k)$ whose dynamics are governed by (6) such that the variables $x_1, \ldots, x_k$ identify with the state of some unspecified and nondimensionalized scalar atmospheric quantity (e.g. wind velocity, moisture) in K sectors of a latitude circle [29]. The dynamics for x_1 require definitions for x_0 and x_{-1}, which are given as $x_0 = x_K$ and $x_{-1} = x_{K-1}$. x_K also necessitates the definition for x_{K+1} given as $x_{K+1} = x_1$ which leads to the following boundary states for x_1 and x_K:

$$\begin{aligned}\dot{x}_1(t) &= -x_{K-1}(t)x_K(t) + x_K(t)x_2(t) - x_1(t) + F \\ \dot{x}_K(t) &= -x_{K-2}(t)x_{K-1}(t) + x_{K-1}(t)x_1(t) - x_K(t) + F\end{aligned} \tag{21}$$

The linear terms in (6) and (21) represent internal dissipation, whereas the quadratic terms simulate advection [23]. As discussed in Sec.(II-B), for small values of F, the system has a steady-state solution: $x_1 = \ldots = x_k = F$. For larger values of F, solutions can exhibit periodicity, bifurcations, and even chaos for state-spaces $N \geq 4$. For our case, solutions corresponding to $(N, F) = (4, 9)$ [branch point], $(N, F) = (10, 4)$ [period doubling], and $(N, F) = (30, 7)$ [chaos] are explored with various levels of parallelization. It should be noted that the integration accuracy thresholds for chaotic systems were set several orders of magnitude lower than the estimation accuracy of the QoI to avoid numerical inconsistencies throughout estimation timeline. Time has also been normalized so that one unit represents 5 days in the Lorenz-96 system. The QoI is set to be the mean value of the state ($h(\mathbf{x}_t = \mathbb{E}[\mathbf{x}_t]$), which results in the cumulative root-mean-square error triggering ensemble adaptations.

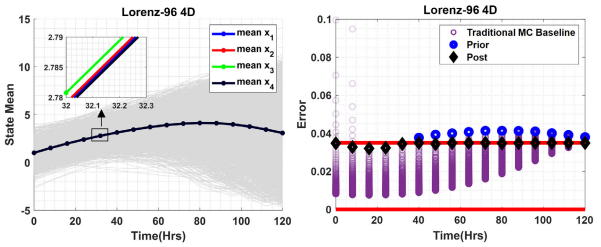

FIGURE 12. Lorenz-96 4D: QoI.

The initial uncertainty is defined as $\mathcal{W}_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ where $\mu_0 = [1, 1, 1, \ldots, 1, 1] \in \mathbb{R}^K$ and $\Sigma_0 = \mathbf{1}_K$ (identity matrix of size $K \times K$). For each case, the lower and upper estimation error bounds were set to $\mathcal{E}_t^{L^*} = 0$ (ensemble thinner turned off) and $\mathcal{E}_t^{U^*} = 0.035$ for consistency across the increasing state-space dimension.

Similar to Fig.(10), Fig.(12) shows the time history of the QoI (mean value of the state) on the left hand side along with required ensembles adaptations to meet the prescribed error bounds on the right. As time progresses, the state mean varies from $\approx$ 1 unit to a maximum of 4.1 units at 80 hrs. with instances of the state uncertainty underlayed in gray. The evolution of the QoI estimation error for both the traditional parallel BRS-MC simulation and AMC is once again shown on the right hand side of Fig.(12). As $(N, F) = (4, 9)$ is the second branch point for the Lorenz-96 model in $4D$, the state mean becomes an increasingly difficult quantity to track throughout the forecast. This can be visualized via the blue circles in Fig.(12), which denote required AMC ensemble adaptations starting at $t = 40$ hrs. to $t_f = 120$ hrs. Correspondingly, the parallel BRS-MC error evolution is also shown on the right hand side of the figure, which was determined in an identical way to the EDL simulation. That is, the parallel BRS-MC simulation was restarted with an additional batch of particles any time the prescribed $\mathcal{E}_t^{U^*}$ was violated. It should be noted that AMC enriches the ensemble at every time instance from $t = 40$ to $t_f = 120$ hrs, meaning the parallel BRS-MC simulation has to be reinitialized numerous times to eventually meet the required accuracy bounds at t_f. In terms of simulation time, this should be thought of as a near worst-case scenario for BRS-MC. That is, comparing Fig.(10) to Fig.(12), one can see that having to restart the parallel BRS-MC simulation to meet $\mathcal{E}_t^{U^*}$ subsided after $t = 4$ min. in EDL, whereas it continued until t_f for Lorenz-96 in $4D$. The overall simulation times for parallel BRS-MC and AMC are further visualized in Fig.(13).

The computational time breakdown for the $4D$ Lorenz-96 model as it relates to the AMC platform and parallel BRS-MC simulation (utilizing the optimal number of cores) is shown on the left hand side of Fig.(13). The overall simulation time for parallel BRS-MC with 500 and 100 batch particle additions are shown by the dashed green and purple lines. For the AMC platform, the number of parallel cores is swept from $1 - 10$ cores, with the overall minimum simulation time ($t_{\min} = 41.91$ min.) occurring at 8 CPU cores. More so, module 3 and module 4 in AMC (Fig.(2)) have

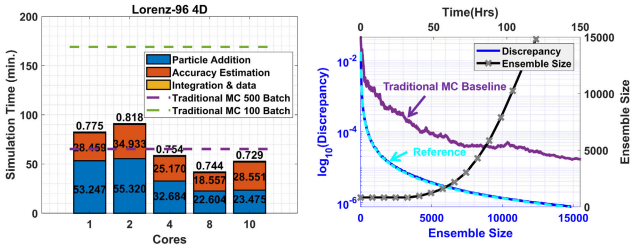

FIGURE 13. Lorenz-96 4D: Simulation Time, Ensemble Size, and Discrepancy.

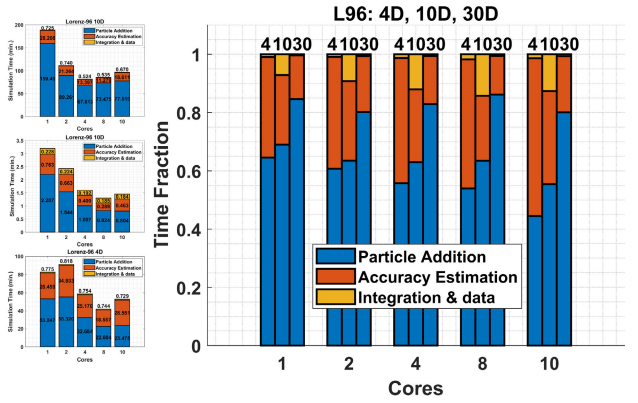

FIGURE 14. Lorenz-96: Parallelization Comparison.

been parallelized with the overall minimum simulation time occurring at the minimums of each individual component. Like EDL, there is significant time savings for the larger batch parallel BRS-MC simulation. The right hand side of Fig.(13) depicts the evolution of (14) (discrepancy) for the 4D Lorenz-96 model alongside the would-be discrepancy of parallel BRS-MC and time evolution of the AMC ensemble size shown via the black axis. Consistent with the EDL scenario, the discrepancy of the parallel BRS-MC simulation is several orders of magnitude above AMC. This is an expected result as module 3 (ensemble enhancer) purposefully minimizes (14) through the execution of SA whereas parallel BRS-MC circumvents this through an alternate sampling routine. The exponential growth of the ensemble size from 825 to 14801 particles over time (shown in black) corresponds to the violations of $\mathcal{E}_t^{U^*}$ in Fig.(12). Upon violating $\mathcal{E}_t^{U^*}$, the ensemble enhancer is triggered for enrichment and performance is reevaluated which ultimately contributes to the overall simulation times shown on the left of Fig.(13).

Additional analysis of the AMC simulation time associated with the Lorenz-96 model is shown in Fig.(14). Overall run times for $(N, F) = (4, 9)$ [branch point], $(N, F) = (10, 4)$ [period doubling], and $(N, F) = (30, 7)$ [chaos] are shown minimized on the left hand side. Run times vary from a maximum of 91.07 min. (4D), 3.19 min. (10D), 188.42 min. (30D) with the disparity in simulation time being largely attributed to the ease of estimating the QoI given $\mathcal{E}_t^{U^*} = 0.035$ versus system behavior. That is, the error on $h(\mathbf{x}_t) = \mathbb{E}[\mathbf{x}_t]$ is significantly easier to control in the 10D period doubling simulation as opposed to the branch point or chaotic systems. The right hand side of Fig.(14) normalizes the simulation times for each of the 4D, 10D, and 30D parallel AMC Lorenz-96 cases to determine the fraction of time consumed by each module. Regardless of the number of cores, the ensemble enhancer routine consumes $60-80\%+$ of the overall run time for each simulation. As the dimension of the state-space increases in Lorenz-96 model, the clock time fraction for ensemble enrichment increases alongside it. This can be partially attributed to the phenomena shown in Fig.(4) with discussions in Sec.(IV-A). That is, the strict error bound of $\mathcal{E}_t^{U^*} = 0.035$ for a chaotic Lorenz-96 in 30D, quickly pushes the ensemble size over the computational time jump shown in the figures, thereby inherently costing more clock time for the same $\mathcal{E}_t^{U^*}$ bound of a lower dimensional system. In fact, the ending ensemble sizes of the optimal AMC platform for Lorenz-96 in 4D, 10D, and 30D are: 14815, 1505, and 7175, reinforcing the observations of Fig.(4) that higher dimensional systems experience a clock time increase when minimizing (14) due to a combination of ensemble and state-space sizes.

An increase in overall simulation time can also be observed for each scenario when moving from 8 (4 in 30D) cores to 10 (8 in 30D). Referring back to Sec.(V-C), parallel SA and task decomposition are balanced through a resynchronization step. That is, the ensemble enhancer is decomposed across CPU cores to balance the computational load as much as possible. However, it is possible that CPU_1 completes its temperature cycle quicker than CPU_2 within SA. This can be traced back to the transition proposal on line 3 of Alg.(2), where candidate particles are sampled from a proposal density (e.g. if the sampled candidate is outside the domain of (14) (discrepancy) resampling is required). The resynchronization step shown in Fig.(8), forces SA to remain in parallel at the start of each temperature cycle, but this may lead to additional core idling as the number of parallel CPUs is increased. Combining this observation with that discussed above in relation to higher dimenensional systems, one can observe the optimal number of utilized CPUs is 8 (lower dimensional systems) or 4 when $\mathcal{E}_t^{U^*} = 0.035$, with 4 and 8 CPUs completing within 4 mins. ($\sim$ 80 min. total clock time) of each other for Lorenz-96 in 30D.

C. COMPETITIVE LOTKA-VOLTERRA MODEL

Introduced in Sec.(II-C), the Lotka-Volterra dynamics also require definitions for both a_{ij} and r_i. For this article, we set $a_{ij} \geq 0$, which corresponds with species competition and we invoke $r_i > 0$ to ensure that all solutions with non-negative initial conditions remain bounded. The lower dimensional (4D) Lokta-Volterra case is based off the chaotic system determined by Vano et. al. and repeated below [34]:

$$r_i = \begin{bmatrix} 1 \\ 0.72 \\ 1.53 \\ 1.27 \end{bmatrix}, \quad a_{ij} = \begin{bmatrix} 1 & 1.09 & 1.52 & 0 \\ 0 & 1 & 0.44 & 1.36 \\ 2.33 & 0 & 1 & 0.47 \\ 1.21 & 0.51 & 0.35 & 1 \end{bmatrix} \tag{22}$$

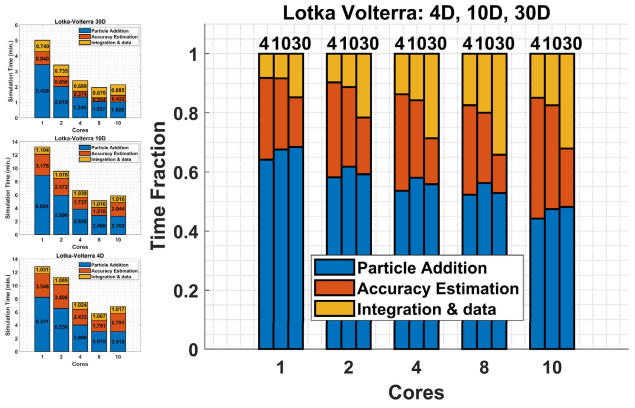

FIGURE 15. Lotka-Volterra: Parallelization Comparison.

The higher dimensional Lotka-Volterra systems (10D and 30D) build off (22) by randomly generating additional a_{ij} and r_i values while still invoking the $a_{ij} \geq 0$ and $r_i > 0$ constraints. Once again, it should be noted that the integration accuracy thresholds for chaotic systems were set several orders of magnitude lower than the estimation accuracy of the QoI. Similar to the Lorenz-96 test cases, the QoI for Lotka-Volterra is set to be the mean value of the state, $h(\mathbf{x}_t) = \mathbb{E}[\mathbf{x}_t]$. The initial uncertainty is defined as $\mathcal{W}_0 \sim \mathcal{N}(\mu_0, \Sigma_0)$ where $\mu_0 = [\mu_{01}, \mu_{02}, \mu_{03}, \ldots, \mu_{0k-1}, \mu_{0k}] \in \mathbb{R}^K$ and $\Sigma_0 = 0.01 * \mathbf{1}_K$ (identity matrix of size $K \times K$). For the 4D case, $\mu_0 = [0.1, 0.35, 0.25, 0.4]$. For each case, the lower and upper estimation error bounds were set to $\mathcal{E}_t^{L^*} = 0$ (ensemble thinner turned off) and $\mathcal{E}_t^{U^*} = 0.004$ for consistency across the increasing state-space dimension.

Consistent with the EDL and Lorenz-96 studies, the run times for the 4D, 10D, and 30D Lotka-Volterra systems are shown minimized on the left hand side of Fig.(15). Run times range from a maximum of 12.83 min. (4D), 13.21 min. (10D), and 5.01 min. (30D) with 1 CPU to a minimum of 5.78 min. (4D), 5.09 min. (10D), and 1.96 min. (30D) with 8 CPUs. It should be noted that maximum and minimum parallel BRS-MC simulation times for 30D Lotka-Volterra were also determined to be 4.5 min. (100 batch) and 1.92 min. (500 batch) with the latter being nearly identical AMC in 30D.

The right hand side of Fig.(15) once again normalizes the simulation times for each of the 4D, 10D, and 30D parallel AMC Lotka-Volterra cases to determine the fraction of time consumed by each module. In slight contrast to Lorenz-96, the ensemble enhancer for Lotka-Volterra consumes approximately $50-60\%$ of the overall simulation clock time. While still significant, this means the ensemble enhancer in Lotka-Volterra did not experience the computational time jump in computing discrepancy for high dimensional systems or large ensemble sizes. In fact, the ending ensemble sizes for the 4D, 10D, and 30D Lotka-Volterra systems are: 4435, 3525, and 1275, which are all well below the time jumps depicted in Sec.(IV-A). It should be noted that the integration of the system dynamics consumed a substantial fraction of clock time for Lotka-Volterra when compared to Lorenz-96. As this is a characteristic of the inherit system, the time fraction cannot be significantly reduced. Parallelization routines and alternative numerical integration schemes may be able to be implemented. However, the difficulty in integrating the system dynamics is a characteristic of the governing equations themselves. In general, the overall minimum simulation time for Lotka-Volterra occurs at 8 CPUs in all scenarios. This also corresponds to the minimum clock times for the ensemble enhancer as well as the accuracy estimation routines in each case.

VII. CONCLUSION

The AMC platform was re-engineered to include the novel implementation of a parallel global stochastic optimization routine alongside additional module enhancements such as arithmetic minimization. The optimization and parallelization of a sequence of cost functions within the ensemble enhancer of the AMC platform significantly reduced the overall simulation time required to generate timely, trustworthy, and accurate forecasts for low and high dimensional systems alike. It has been shown that the parallel AMC platform consumes 248.7% less clock time for EDL, 403.42% for 4D Lorenz-96, and and 229.47% for 30D Lotka-Volterra when compared to BRS-MC with 100 batch particle additions with similar savings in 10D and 30D (4D for Lotka-Volterra). It was observed that the ensemble enhancer consumes a significant fraction of the total simulation time regardless of the system dynamics or state space dimension. While this is an expected result as to meet the required accuracy bounds, it should be noted that the ensemble enhancer consumes a larger time fraction for high-dimensional systems with considerable ensemble sizes. This is mainly due to the computational time jump associated with optimizing discrepancy in high dimensional spaces with large ensembles. More so, the optimal number of parallel CPU cores for most scenarios was found to be 8 due to the balancing of task decomposition and the resynchronization of parallel SA. Moving forward, alternatives to SA such as the momentum method and Nesterov's accelerated gradient, among other various global optimization techniques will be explored to minimize resynchronization while potentially reducing the overall computational effort. The addition of m particles per CPU as opposed to a *single* particle is also a worthwhile avenue of exploration. The implementation of single precision can save significant clock time and will be introduced as the AMC platform continues to be restructured and evolved for efficient forecasting. Pertinent studies related to the efficacy and efficiency of the performance evaluator are also planned for future development.

ACKNOWLEDGMENT

The authors would like to thank Dr. Michael Yakes of the Air Force Office of Scientific Research for his thoughtful technical support.

REFERENCES

[1] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo Method* (Wiley Series in Probability and Statistics), 3rd ed. Hoboken, NJ, USA: Wiley, 2017.

[2] C. Yang, "On particle methods for uncertainty quantification in complex systems," Ph.D. dissertation, Dept. Mech. Aerosp. Eng., Ohio State Univ., Columbus, OH, USA, 2017.

[3] R. E. Bellman, *Dynamic Programming* (Princeton Landmarks in Mathematics and Physics), 1st ed. Princeton, NJ, USA: Princeton Univ. Press, 2010.

[4] M. Kumar, "Design and analysis of stochastic dynamical systems with fokker-planck equation," Ph.D. dissertation, Texas A&M Univ. Libraries, College Station, TX, USA, Feb. 2011. [Online]. Available: https://hdl.handle.net/1969.1/ETD-TAMU-2009-12-7500

[5] M. Kumar, S. Chakravorty, and J. L. Junkins, "A semianalytic meshless approach to the transient Fokker–Planck equation," *Probabilistic Eng. Mech.*, vol. 25, no. 3, pp. 323–331, Jul. 2010.

[6] M. Kumar, S. Chakravorty, P. Singla, and J. L. Junkins, "The partition of unity finite element approach with hp-refinement for the stationary Fokker–Planck equation," *J. Sound Vibrat.*, vol. 327, nos. 1–2, pp. 144–162, Oct. 2009.

[7] M. Kumar, S. Chakravortyt, and J. L. Junkins, "On the curse of dimensionality in the fokker-planck equation," *Adv. Astron. Sci.*, vol. 135, no. 3, pp. 1781–1800, 2009.

[8] Y. Sun and M. Kumar, "A meshless p-PUFEM Fokker-Planck equation solver with automatic boundary condition enforcement," in *Proc. Amer. Control Conf. (ACC)*, Jun. 2012, pp. 74–79.

[9] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[10] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings, "A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modeling of complex fluids," *J. Non-Newtonian Fluid Mech.*, vol. 139, no. 3, pp. 153–176, Dec. 2006.

[11] A. Ammar, B. Mokdad, F. Chinesta, and R. Keunings, "A new family of solvers for some classes of multidimensional partial differential equations encountered in kinetic theory modelling of complex fluids: Part II: Transient simulation using space-time separated representations," *J. Non-Newtonian Fluid Mech.*, vol. 144, pp. 98–121, Jul. 2007.

[12] F. Chinesta, A. Ammar, and E. Cueto, "Proper generalized decomposition of multiscale models," *Int. J. Numer. Methods Eng.*, vol. 83, nos. 8–9, pp. 1114–1132, Aug. 2010.

[13] F. Chinesta, A. Ammar, and E. Cueto, "Recent advances and new challenges in the use of the proper generalized decomposition for solving multidimensional models," *Arch. Comput. Methods Eng.*, vol. 17, no. 4, pp. 327–350, 2010.

[14] M. B. Giles, "Multilevel Monte Carlo path simulation," *Oper. Res.*, vol. 56, no. 3, pp. 607–617, 2008.

[15] M. Valli, R. Armellin, P. Di Lizia, and M. Lavagna, "A Gaussian particle filter based on differential algebra for spacecraft navigation," Int. Astron. Fed., Naples, Italy, Tech. Rep., 2012, pp. 5178–5187, vol. 7.

[16] B. Peherstorfer, K. Willcox, and M. Gunzburger, "Optimal model management for multifidelity Monte Carlo estimation," *SIAM J. Sci. Comput.*, vol. 38, no. 5, pp. A3163–A3194, Jan. 2016.

[17] P. D. Moral, A. Jasra, K. J. H. Law, and Y. Zhou, "Multilevel sequential Monte Carlo samplers for normalizing constants," *ACM Trans. Model. Comput. Simul.*, vol. 27, no. 3, pp. 1–22, Aug. 2017.

[18] A. A. Gorodetsky, G. Geraci, M. S. Eldred, and J. D. Jakeman, "A generalized approximate control variate framework for multifidelity uncertainty quantification," *J. Comput. Phys.*, vol. 408, May 2020, Art. no. 109257.

[19] J. Warner, S. C. Niemoeller, L. Morrill, G. Bomarito, P. Leser, W. Leser, R. A. Williams, and S. Dutta, *Multi-Model Monte Carlo Estimators for Trajectory Simulation*. Reston, VA, USA: AIAA, 2021.

[20] G. F. Bomarito, P. E. Leser, J. E. Warner, and W. P. Leser, "On the optimization of approximate control variates with parametrically defined estimators," *J. Comput. Phys.*, vol. 451, Feb. 2022, Art. no. 110882.

[21] J. Konrad, I.-G. Farcaş, B. Peherstorfer, A. Di Siena, F. Jenko, T. Neckel, and H.-J. Bungartz, "Data-driven low-fidelity models for multi-fidelity Monte Carlo sampling in plasma micro-turbulence analysis," *J. Comput. Phys.*, vol. 451, Feb. 2022, Art. no. 110898.

[22] C. Yang and M. Kumar, "On the effectiveness of Monte Carlo for initial uncertainty forecasting in nonlinear dynamical systems," *Automatica*, vol. 87, pp. 301–309, Jan. 2018.

[23] C. Yang and M. Kumar, "Closed-loop adaptive Monte Carlo framework for uncertainty forecasting in nonlinear dynamic systems," *J. Guid., Control, Dyn.*, vol. 42, no. 6, pp. 1218–1236, Jun. 2019.

[24] K. Crombecq, E. Laermans, and T. Dhaene, "Efficient space-filling and non-collapsing sequential design strategies for simulation-based modeling," *Eur. J. Oper. Res.*, vol. 214, no. 3, pp. 683–696, Nov. 2011.

[25] B. Øksendal, *Stochastic Differential Equations: An Introduction With Applications* (Universitext), 6th ed. Berlin, Germany: Springer-Verlag, 2003.

[26] C. Yang and M. Kumar, "An adaptive Monte Carlo method for uncertainty forecasting in perturbed two-body dynamics," *Acta Astronautica*, vol. 155, pp. 369–378, Feb. 2019.

[27] N. X. Vinh, A. Busemann, and R. D. Culp, *Hypersonic and Planetary Entry Flight Mechanics*. Ann Arbor, MI, USA: Univ. Michigan Press, 1990.

[28] A. W. VanFossen and M. Kumar, *Parallelized Global Stochastic Optimization for Efficient Ensemble Enhancement Within an Adaptive Monte Carlo Forecasting Platform*. Reston, VA, USA: AIAA, 2021.

[29] E. Lorenz, "Predictability: A problem partly solved," in *Seminar on Predictability*, vol. 1. Shinfield Park, U.K.: ECMWF, 1995, pp. 1–18. [Online]. Available: https://www.ecmwf.int/node/10829

[30] D. van Kekem, "Dynamics of the lorenz-96 model: Bifurcations, symmetries and waves," Ph.D. dissertation, Dept. Sci. Eng., Univ. Groningen, Groningen, The Netherlands, 2018.

[31] E. N. Lorenz, "Irregularity: A fundamental property of the atmosphere," *Tellus A*, vol. 36A, no. 2, pp. 98–110, Mar. 1984.

[32] D. Orrell, L. Smith, J. Barkmeijer, and T. N. Palmer, "Model error in weather forecasting," *Nonlinear Processes Geophys.*, vol. 8, no. 6, pp. 357–371, Dec. 2001.

[33] E. Ott, B. R. Hunt, I. Szunyogh, A. V. Zimin, E. J. Kostelich, M. Corazza, E. Kalnay, D. Patil, and J. A. Yorke, "A local ensemble Kalman filter for atmospheric data assimilation," *Tellus A, Dyn. Meteorol. Oceanogr.*, vol. 56, no. 5, pp. 415–428, 2004.

[34] J. A. Vano, J. C. Wildenberg, M. B. Anderson, J. K. Noel, and J. C. Sprott, "Chaos in low-dimensional Lotka–Volterra models of competition," *Nonlinearity*, vol. 19, no. 10, pp. 2391–2404, Sep. 2006.

[35] M. W. Hirsch, "Systems of differential equations which are competitive or cooperative: I. Limit sets," *SIAM J. Math. Anal.*, vol. 13, no. 2, pp. 167–179, Mar. 1982.

[36] M. W. Hirsch, "Systems of differential equations that are competitive or cooperative II: Convergence almost everywhere," *SIAM J. Math. Anal.*, vol. 16, no. 3, pp. 423–439, 1985.

[37] S. Smale, "On the differential equations of species in competition," *J. Math. Biol.*, vol. 3, no. 1, pp. 5–7, 1976.

[38] C. Yang and M. Kumar, "Discrepancy driven adaptive Monte Carlo for forward uncertainty forecasting in nonlinear dynamical systems," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 5448–5453.

[39] H. Niederreiter, "Quasi-Monte Carlo methods and pseudo-random numbers," *Bull. Amer. Math. Soc.*, vol. 84, no. 6, pp. 957–1041, 1978.

[40] H. Janssen, "Monte-Carlo based uncertainty analysis: Sampling efficiency and sampling convergence," *Rel. Eng. Syst. Safety*, vol. 109, pp. 123–132, Jan. 2013.

[41] A. Grosso, A. R. M. J. U. Jamali, and M. Locatelli, "Finding maximin Latin hypercube designs by iterated local search heuristics," *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 541–547, Sep. 2009.

[42] T. W. Simpson, J. Peplinski, P. N. Koch, and J. K. Allen, "Metamodels for computer-based engineering design: Survey and recommendations," *Eng. Comput.*, vol. 17, no. 2, pp. 129–150, Jul. 2001.

[43] K.-T. Fang and D. K. Lin, "Uniform experimental designs and their applications in industry," in *Statistics in Industry* (Handbook of Statistics), vol. 22. Amsterdam, The Netherlands: Elsevier, 2003, pp. 131–170, ch. 4.

[44] F. Hickernell, "A generalized discrepancy and quadrature error bound," *Math. Comput. Amer. Math. Soc.*, vol. 67, no. 221, pp. 299–322, 1998.

[45] J. C. Spall, *Annealing-Type Algorithms*. Hoboken, NJ, USA: Wiley, 2003, pp. 208–230, doi: 10.1002/0471722138.ch8.

[46] B. Hajek, "Cooling schedules for optimal annealing," *Math. Oper. Res.*, vol. 13, no. 2, pp. 311–329, 1988. [Online]. Available: http://www.jstor.org/stable/3689827

[47] C. Yang and M. Kumar, "An information guided framework for simulated annealing," *J. Global Optim.*, vol. 62, no. 1, pp. 131–154, May 2015.

[48] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[49] L. Ingber, "Simulated annealing: Practice versus theory," *Math. Comput. Model.*, vol. 18, no. 11, pp. 29–57, 1993. [Online]. Available: https://www.sciencedirect.com/science/article/pii/089571779390204C

[50] M. McCool, A. D. Robison, and J. Reinders, "Chapter 1—Introduction," in *Structured Parallel Programming*. Boston, MA, USA: Morgan Kaufmann, 2012, pp. 1–38.
[51] M. McCool, A. D. Robison, and J. Reinders, "Chapter 2—Background," in *Structured Parallel Programming*. Boston, MA, USA: Morgan Kaufmann, 2012, pp. 39–75.
[52] K. Binder and D. Stauffer, *A Simple Introduction to Monte Carlo Simulation and Some Specialized Topics*. Berlin, Germany: Springer, 1987, pp. 1–36.
[53] Q. Wu and T. Sloane, "CMOS leaf-cell design using simulated annealing," in *Proc. Midwest Symp. Circuits Syst.*, Washington, DC, USA, 1992, pp. 1516–1519, vol. 2.
[54] R. S. Bucy and R. S. Diesposti, "Decision tree design by simulated annealing," *ESAIM, Math. Model. Numer. Anal.*, vol. 27, no. 5, pp. 515–534, 1993. [Online]. Available: http://eudml.org/doc/193713
[55] R. W. Klein and R. C. Dubes, "Experiments in projection and clustering by simulated annealing," *Pattern Recognit.*, vol. 22, no. 2, pp. 213–220, Jan. 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0031320389900678
[56] H. Raittinen and K. Kaski, "Image deconvolution with simulated annealing method," *Phys. Scripta*, vol. T33, pp. 126–130, Jan. 1990.
[57] D. H. Ackley, G. H. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognit. Sci.*, vol. 9, pp. 147–169, Jan. 1985. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0364021385800124
[58] A. V. Lukashin, J. Engelbrecht, and S. Brunak, "Multiple alignment using simulated annealing: Branch point definition in human mRNA splicing," *Nucleic Acids Res.*, vol. 20, no. 10, pp. 2511–2516, 1992.
[59] T. L. Beck, J. D. Doll, and D. L. Freeman, "Locating stationary paths in functional integrals: An optimization method utilizing the stationary phase Monte Carlo sampling function," *J. Chem. Phys.*, vol. 90, no. 6, pp. 3181–3191, Mar. 1989.
[60] L. Ingber, "Statistical-mechanical aids to calculating term-structure models," *Phys. Rev. A, Gen. Phys.*, vol. 42, no. 12, pp. 7057–7064, Dec. 1990, doi: 10.1103/PhysRevA.42.7057.
[61] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein, "Generalized simulated annealing for function optimization," *Technometrics*, vol. 28, no. 3, pp. 209–217, Aug. 1986.
[62] G. Bilbro, R. Mann, T. Miller, W. Snyder, D. van den Bout, and M. White, "Optimization by mean field annealing," in *Advances in Neural Information Processing Systems*, vol. 1, D. Touretzky, Ed. San Mateo, CA, USA: Morgan Kaufmann, 1988.
[63] E. Marinari and G. Parisi, "Simulated tempering: A new Monte Carlo scheme," *Europhys. Lett.*, vol. 19, no. 6, p. 451, Jul. 1992.
[64] H. Szu and R. Hartley, "Fast simulated annealing," *Phys. Lett. A*, vol. 122, nos. 3–4, pp. 157–162, Jun. 1987.
[65] S. P. Brooks and B. J. Morgan, "Optimization using simulated annealing," *Statistician*, vol. 44, no. 2, pp. 241–257, 1995. [Online]. Available: http://www.jstor.org/stable/2348448
[66] P. Salamon, P. Sibani, and R. Frost, *Conjectures, and Improvements for Simulated Annealing* (Society for Industrial and Applied Mathematics). Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.
[67] K. H. Hoffmann, A. Franz, and P. Salamon, "Structure of best possible strategies for finding ground states," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 66, no. 4, Oct. 2002, Art. no. 046706, doi: 10.1103/PhysRevE.66.046706.
[68] B. Andresen and J. M. Gordon, "Constant thermodynamic speed for minimizing entropy production in thermodynamic processes and simulated annealing," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 50, no. 6, pp. 4346–4351, Dec. 1994, doi: 10.1103/PhysRevE.50.4346.
[69] E. Onbaşioglu and L. Özdamar, "Parallel simulated annealing algorithms in global optimization," *J. Global Optim.*, vol. 19, pp. 27–50, 2001.
[70] S.-Y. Lee and K. G. Lee, "Synchronous and asynchronous parallel simulated annealing with multiple Markov chains," *IEEE Trans. Parallel Distrib. Syst.*, vol. 7, no. 10, pp. 993–1008, Oct. 1996.
[71] G. Kliewer and S. Tschoke, "A general parallel simulated annealing library and its application in airline industry," in *Proc. 14th Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2000, pp. 55–61.
[72] L. Özdamar and M. Demirhan, "Experiments with new stochastic global optimization search techniques," *Comput. Oper. Res.*, vol. 27, no. 9, pp. 841–865, Aug. 2000.
[73] Y.-Z. Luo and G.-J. Tang, "Parallel simulated annealing using simplex method," *AIAA J.*, vol. 44, no. 12, pp. 3143–3146, Dec. 2006.
[74] Z. Lou and J. Reinitz, "Parallel simulated annealing using an adaptive resampling interval," *Parallel Comput.*, vol. 53, pp. 23–31, Apr. 2016.
[75] T. Ribeiro, J. Rufino, and A. I. Pereira, "PSSA: Parallel stretched simulated annealing," in *Proc. AIP Conf.*, vol. 1389, no. 1, 2011, pp. 783–786, doi: 10.1063/1.3636849.
[76] S. Lee and S. B. Kim, "Parallel simulated annealing with a greedy algorithm for Bayesian network structure learning," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 6, pp. 1157–1166, Jun. 2020.
[77] K. Gonçalves-e-Silva, D. Aloise, and S. Xavier-de-Souza, "Parallel synchronous and asynchronous coupled simulated annealing," *J. Supercomput.*, vol. 74, no. 6, pp. 2841–2869, Jun. 2018.
[78] A. W. VanFossen and M. Kumar, "Determination of efficient quantities of interest for space situational awareness and adaptive Monte Carlo," in *Proc. 31st AAS/AIAA Space Flight Mech. Meeting*, 2021, pp. 391–402.
[79] A. Halder and R. Bhattacharya, "Dispersion analysis in hypersonic flight during planetary entry using stochastic Liouville equation," *J. Guid., Control, Dyn.*, vol. 34, no. 2, pp. 459–474, Mar. 2011.
[80] K. Edquist, A. Dyakonov, M. Wright, and C. Tang, "Aerothermodynamic design of the Mars science laboratory heatshield," in *Proc. 41st AIAA Thermophys. Conf.*, Jun. 2009, pp. 2009–4075.

ANDREW W. VANFOSSEN received the B.S. degree in aerospace engineering from The Ohio State University, Columbus, OH, USA, in 2017, where he is currently pursuing the Ph.D. degree in aerospace engineering. From 2017 to 2019, he was a Guidance, Navigation, and Control Engineer at Raytheon Technologies working in conjunction with the United States Navy on projects, such as raid annihilation and trajectory shaping. Since 2019, he has been a Research Assistant with the Laboratory for Autonomy in Data-Driven and Complex Systems, Mechanical and Aerospace Engineering Department, The Ohio State University. His research interests include machine learning, uncertainty forecasting, autonomy, and numerical methods.

Mr. Vanfossen's awards and honors include the University Fellowship and the Discovery Scholars Fellowship from The Ohio State University as well as a finalist for the Guidance, Navigation and Control Best Graduate Student Paper Competition at AIAA SciTech.

MRINAL KUMAR (Member, IEEE) received the B.Tech. degree in aerospace engineering from IIT Kanpur, in 2004, and the Ph.D. degree in aerospace engineering from Texas A&M University, College Station, TX, USA, in 2009.

He is the Elizabeth Martin Tinkham Endowed Professor of aeronautical and astronautical engineering with The Ohio State University (OSU), where he is the Director of the Laboratory for Autonomy in Data-Driven and Complex Systems, Aerospace Research Center. His research interests include uncertainty forecasting and dynamics learning for space domain awareness and mission planning and execution for small unmanned aerial systems (sUAS) in wildland fires. He is an Associate Fellow of AIAA.